6-16-2000

The complexity of planning with partially-observable Markov decision processes

Martin Mundhenk
Dartmouth College

The Complexity of Planning with Partially-Observable Markov Decision Processes

Martin Mundhenk*

June 16, 2000

Abstract

This work surveys results on the complexity of planning under uncertainty. The planning model considered is the partially-observable Markov decision process. The general planning problems are, given such a process, (a) to calculate its performance under a given control policy, (b) to find an optimal or approximate optimal control policy, and (c) to decide whether a good policy exists. The complexity of this and related problems depend on a variety of factors, including the observability of the process state, the compactness of the process representation, the type of policy, or even the number of actions relative to the number of states. In most cases, the problem can be shown to be complete for some known complexity class.

The skeleton of this survey are results from Littman, Goldsmith, and Mundhenk (1998), Mundhenk (2000), Mundhenk, Goldsmith, Lusena, and Allender (2000), and Lusena, Goldsmith, and Mundhenk (1998). But there are also some news.

1 Introduction

Some argue that it is the nature of man to attempt to control his environment. However, control policies, whether of stock market portfolios or the water supply for a city, sometimes fail. One problem in controlling complex systems is that the actions chosen rarely have determined outcomes: how the stock market or the weather behaves is something that we can, at best, model probabilistically. Another problem in many controlled stochastic systems is that the sheer size of the system makes it infeasible to thoroughly test or even analyze alternatives. Furthermore, a controller must take into account the costs as well as rewards associated with the actions chosen. This work considers the computational complexity of the basic tasks facing such a controller.

We investigate here a mathematical model of controlled stochastic systems with utility called *partially-observable Markov decision process* (POMDP), where the exact state of the system may at times be hard or impossible to recognize. We consider problems of interest when a controller has been given a POMDP modeling some system. We assume that the system has been described in terms of a finite state space and a finite set of actions, with full information about the probability distributions and rewards associated with being in each state and taking each action. Given such a model, a controller needs to be able to find and evaluate control policies. We investigate the computational complexity of these problems. Unsurprisingly to either algorithm designers or theorists, we show that incomplete information about the state of the system hinders computation. However, the patterns formed by our results are not as regular as one might expect (see Tables 1–3).

*Supported in part by the Deutsche Akademische Austauschdienst (DAAD) grant 315/PPP/gü-ab

In Operations Research, POMDPs are usually assumed to be continuous. We do not address any of the complexity issues for the continuous case here. In addition, there is a large literature on acquiring the actual POMDPs, usually by some form of automated learning or by knowledge elicitation. While there are fascinating questions of algorithms, complexity and psychology (see for instance van der Gaag, Renooij, Witteman, Aleman, and Tall (1999)) associated with these processes, we do not address them here.

In the remainder of this section, we discuss the basic models of Markov decision processes, we consider the most important parameters that affect the complexity of computational problems for POMDPs, and we present an overview of the rest of this work.

Markov Decision Processes

Markov decision processes model sequential decision-making problems. At a specified point in time, a decision maker observes the state of a system and chooses an action. The action choice and the state produce two results: the decision maker receives an immediate reward (or incurs an immediate cost), and the system evolves probabilistically to a new state at a subsequent discrete point in time. At this subsequent point in time, the decision maker faces a similar problem. The observation made from the system's state now may be different from the previous observation; the decision maker may use the knowledge about the history of the observations. The goal is to find a policy of choosing actions (dependent on the observations of the state and the history) which maximizes the rewards after a certain time.

Bellman (1954) coined the expression "Markov decision process". He investigated *fully-observable* processes – i.e. those which allows to observe the current state exactly. Finding an optimal policy for a fully-observable process is a well studied topic in optimization and control theory (see e.g. Puterman (1994) as one recent book). Tractable algorithmic solutions use dynamic or linear programming techniques. The inherent complexity was first studied in Sondik (1971), Smallwood and Sondik (1973), and Papadimitriou and Tsitsiklis (1987). However, computers are used in more and more areas of daily life, and real life systems lack the convenient property of being fully-observable. In many models, there are properties of a state that are not exactly observable, so the observations only specify a subset of the states to which the current state belongs. The model of *partially-observable* Markov decision processes (abbreviated POMDPs) was introduced by Sondik (1971), who experienced that the dynamic or linear programming techniques used for fully-observable processes are not any more useful for POMDPs. Only for small models can one find exact solutions with feasible resources (see Monahan (1982), Cassandra, Kaelbling, and Littman (1994), Madani, Hanks, and Condon (1999)). Even the known approximations are useful only on systems with tens of states (White (1991), Lovejoy (1991), Cassandra, Kaelbling, and Littman (1995), Cassandra, Littman, and Zhang (1997), Parr and Russell (1995), Zhang and Liu (1997)). Since already simple tasks can need models with thousands of states (see Simmons and Koenig (1995)), the hardness of the problem is well-known. However, despite the work of Papadimitriou and Tsitsiklis (1987) and a few others[1], there were many variants of POMDP problems for which it had not been proved that finding tractable exact solutions or provably good approximations is hard.

We address the computational complexity, given a POMDP and specifications for a decision maker, of (a) evaluating a given policy, (b) finding a good policy for choosing actions, and (c)

[1]There are two recent surveys on the complexity of MDPs and of stochastic control, respectively, that are related to this work. The article by Littman, Dean, and Kaelbling (1995) surveys the complexity of algorithms for fully-observable MDPs; the article by Blondel and Tsitsiklis (2000) discusses both the complexity of algorithms and of the underlying problems, but concentrates on MDPs, too.

finding a near-optimal policy. We abstract the first of these to the *policy evaluation problem*, "Does this policy have expected reward > 0?" The second of these was abstracted by Papadimitriou and Tsitsiklis (1987) to the *policy existence problem*, "Is there a policy with expected reward equal to 0?" The underlying assumption by Papadimitriou and Tsitsiklis (1987) is that all rewards of the process are negative or 0, and thus a policy with reward 0 is optimal (if it exists at all). In contrast, we consider two kinds of processes, (a) processes with only *nonnegative* rewards, and (b) processes with both positive and negative rewards. POMDPs with nonnegative rewards tend to be computationally much simpler, since in this case rewards cannot cancel each other out. We abstract the policy existence problems as follows: "Is there a policy with expected reward > 0?" We have chosen these decision problems because they can be used, along with binary search, to calculate the exact value of the given or the optimal policy[2].

It could be argued that the best way to evaluate a policy is to test it. There are two problems with this approach. One is that it gives us a single possible outcome, rather than a probabilistic analysis. The second is that some systems should *not* be run to test a hypothesis or policy: for instance, a military scenario. Although any POMDP can be simulated, this only gives us an approximation to the expected outcome. Therefore, it is important that we are able to evaluate policies directly.

The complexity of the decision problems depends on specifications on the decision maker and the representation of the process. The specifications on the decision maker include the amount of feedback that the decision maker gets from the system (observability), restrictions on her/his computing resources (type of policy), and the length of time that the process will run (horizon). The straightforward process representation consists of a function for the transition probabilities and a function for the rewards. The usual way is to give these functions as tables. When the system being modeled has sufficient structure, one may use a more concise representation e.g. via circuits, to efficiently specify processes whose table representation would be intractable.

Observability

We consider three models of Markov decision processes: *fully-observable* Markov decision processes, which we refer to simply as MDPs; processes where the decision maker may have incomplete knowledge of the state of the system, called *partially-observable* Markov decision process or POMDPs, and processes where the decision maker has no feedback at all about the present state, which we call *unobservable* Markov decision processes, or UMDPs. (Formal definitions are given in Section 2.) Note that MDPs and UMDPs both are special cases of POMDPs.

The only difference between MDPs and POMDPs is the observability of the system. Whereas with MDPs the exact state of the system can always be observed by the decision maker, this is not the case with POMDPs. The effects of observability on the complexity of calculating an optimal policy for a process are drastic. For MDPs, dynamic programming is used to calculate an optimal policy (for a finite number of decision epochs, called a finite *horizon*) in time polynomial in the size of the system. Moreover, optimal policies can be expressed as functions of the system's current state and the number of previous decision epochs. Papadimitriou and Tsitsiklis (1987) showed that in this case, related decision problems are **P**-complete. In contrast, the optimal policies for POMDPs are generally expressible as functions from initial segments of the history of a process (a series of observations from previous and current times). This is unfortunate because the problem of finding

[2]Binary search involves creating a series of new POMDPs from the given one by appending an initial state such that any action from that state yields a positive or negative reward equal to the shift from the last value considered. It seems that this process requires both negative and positive rewards.

optimal policies for POMDPs is **PSPACE**-hard. Moreover, specifying those policies explicitly may require space exponential in the process description. Hence, it is intractable (again, in the worst case) for a decision maker to choose actions according to an optimal policy for a POMDP.

Policy types and horizons

The computational restrictions of the decision maker are expressed in terms of different types of policies. The simplest policy is called *stationary policy* and takes only the actual observation into account, whereas the *history-dependent policy* makes use of all observations collected during the run-time of the process. A *concise policy* is a history-dependent policy that can briefly be expressed as a circuit. A further restriction is the *time-dependent policy*, which makes its choice of actions dependent on the actual observation and on the number of steps the process passed already. Since UMDPs cannot distinguish histories except as blind sequences of observations, history-dependent policies for UMDPs are equivalent to time-dependent policies.

If one runs a POMDP under a policy, each choice of an action according to the policy has an immediate reward. The expected sum of rewards or of weighted rewards determines the *performance*. For a POMDP there may be different policies with high and with low performances. An *optimal policy* is one with maximal performance. There are different performance-metrics like e.g. total expected sum of rewards, which just adds up the expected rewards, or discounted sum of rewards, which adds the rewards weighted dependent on how many steps the process already run. Whereas the former is the standard metric for processes that run a finite number of steps, the latter guarantees a finite performance even for infinitely running processes.

The horizon is the number of steps (state transitions) a POMDP makes. We distinguish between *short-term* (number of steps equal the size of the description of the POMDP), *long-term* (number of steps exponential in the size of the description of the POMDP), and *infinite* horizon. For infinite horizon MDP policies with discounted rewards, the optimal value is achieved by a stationary policy, and the optimal policy for an MDP with finite horizon is a time-dependent policy (see e.g. Puterman (1994)). For POMDPs, the optimal policy under finite or infinite horizons, total expected, or discounted rewards, is history-dependent.

Representation

A POMDP consists of a finite number of states, a function which maps each state to the observation made from this state, a probabilistic transition relation between states, and a reward function on states; these last two are dependent on actions. The straightforward (or "flat") representation of a POMDP is by a set of tables for the transition relation – one table for each action – and similar tables for the reward function and for the observation function. There is interest in so-called "structured" POMDPs (Bylander (1994), Boutilier, Dearden, and Goldszmidt (1995), Littman (1997a)). These representations arise when one can make use of structures in the state space to provide small descriptions of very large systems. In many cases, when a MDP or POMDP is learned, there is an *a priori* structure imposed on it. For instance, consider the example cited in van der Gaag, Renooij, Witteman, Aleman, and Tall (1999): the system modeled is for diagnosis and treatment of cancer of the esophagus. The actual POMDP has been elicited from physicians. A list of significant substates, or variables, was elicited: characteristics of the carcinoma, depth of invasion into the esophagus wall, state of metastases, etc. An actual state of the system is described by a value for each variable. Then interrelations of the variables and probabilities of transitions were elicited.

There are many ways to model a structured MDP or POMDP, such as two-phase temporal Bayes nets (2TBNs) or probabilistic STRIPS operators (PSOs) (see Blythe (1999) and Boutilier,

Dean, and Hanks (1999) for a discussion of a richer variety of representations); each of these representations may represent a very large system compactly. We choose a different representation that captures the succinctness of all of the representations common in AI, but does not necessarily have the semantic transparency of the others. We use Boolean circuits to describe the computation of probabilities, without specifying any internal representation that reflects the semantics of the situation.

The circuits take as input a state s, an action a, and a potential next state s', and output the probability that action a performed when the system is in state s will take the system to state s'. We call this representation of the transition probability *compressed*. Our results show that compressed representation is not a panacea for "the curse of dimensionality" (Sondik (1971)), although there are certainly instances of compressed represented POMDPs where the optimal policies are easy to find. Thus, future research on efficient policy-finding algorithms for compressed represented POMDPs must focus on structural properties of the POMDPs that limit the structure of the circuits in ways that force our reductions to fail.

In fact, the reductions that we use can be translated into results for a variety of representations. The fact that we use circuits to present these POMDPs is primarily a matter of convenience, for showing that the problems in question are members of their respective complexity classes. It also ties into complexity theory literature on succinct representations (see Galperin and Wigderson (1983), Wagner (1986)).

Applications of POMDPs

MDPs were described by Bellman (1957) and Howard (1960) in the '50's. The first known application of MDPs was to road management in the state of Arizona in 1978 (Golabi, Kulkarni, and Way (1982), as cited in Puterman (1994)). The states represented the amount of cracking of the road surface; the actions were the types of removal, resurfacing, or crack sealing. Puterman also reports on applications of MDPs to wildlife management, factory maintenance, and warehousing and shipping.

In many instances of controlled stochastic systems, the state of the system is not fully-observable. In an industrial or robotic setting, this could be because of sensor failure or simply that sensors cannot distinguish between all the states modeled. For instance, in the robot location problem, the robot can only see its immediate surroundings, which may look just like some other location, at least to the robot. In the organic settings, the controller may simply not have access to full information. For instance, a doctor rarely, if ever, has complete information about a patient's state of health, yet she is called on to make decisions and choose actions based on what she knows, in order to optimize the expected outcome. POMDPs model such systems, and were developed in the '60's by Astrom (1965), Aoki (1965), Dynkin (1965) and Streibel (1965); the final formalization was done by Sondik (1971), and Smallwood and Sondik (1973).

A rich source of information on applications of MDPs and POMDPs is the wealth of recent dissertations, for instance by: Littman (1996), Hauskrecht (1997), Cassandra (1998), Hansen (1998a), and Pyeatt (1999).

Any controlled system represented by Bayesian networks or other similar graphical representations of stochastic systems, as well as any systems represented by probabilistic STRIPS-style operators is an MDP or POMDP. It is beyond the scope of this work to give a full survey of applications of MDPs, POMDPs, and their variants. To give some sense of the breadth of possible applications, we mention those that came up at a recent conference, Uncertainty in AI '99 (Laskey and Prade 1999): medical diagnosis and prediction systems, requirements engineering, intelligent

policy types	representation, horizon, and observability: flat (short-term)	compressed (short-term)	compressed (long-term)
	unobservable		
stationary	**PL**	**PP**	**PSPACE**
time-dependent	**PL**	**PP**	n/a
	fully-observable or partially-observable		
stationary	**PL**	n/a	n/a
time-dependent	**PL**	n/a	n/a
concise	**PP**	**PP**	n/a
history-dependent	**PL**	n/a	n/a

Table 1: The complexity of policy evaluation

buildings, auctions, intelligent e-mail handling and receiver alerting, poker, educational assessment, video advising (choosing the right video to watch tonight), dependability analysis, robot location, and waste management.

While we show conclusively that simply compressing the representation of an MDP or POMDP does not buy more computational power, many of the recent applications have been in the form of these so-called "factored" representations, where different parameters of the states are represented as distinct variables, and actions are represented in terms of the dependence of each variable on others. It has been argued that people do not construct large models without considerable semantic content for the states. One way to represent this content is through such variables. Although there seems to be much more algorithmic development for "flat" or straightforward representations, it seems that there are more examples of systems modeled by compressed or factored representations. Thus, a very fruitful area of research is to develop policy-finding algorithms that take advantage of such factored representations. (For a survey of the state of the art, see Boutilier, Dean, and Hanks (1999).)

Results overview

In Section 4 we consider short-term policy evaluation problems. The main results (see Table 1) show that the general problems are complete for the complexity classes **PL** (probabilistic logarithmic-space) and **PP** (probabilistic polynomial-time). Problems in **PL** can be computed in polynomial time, but more efficiently in parallel than problems in **P** generally can. The problems we show to be **PL**-complete were already known to be in **P** (Papadimitriou and Tsitsiklis (1987)) under a slightly different formulation. Proving them to be **PL**-complete improves the results by Papadimitriou and Tsitsiklis (1987) and completely characterizes the inherent complexity of the problem. This also yields **PSPACE**-completeness for long-term compressed represented POMDPs.

In Section 5, short-term policy existence problems and approximability of the optimal policy and optimal value problems are studied. For policy existence problems (see Tables 2 and 3), in most cases, we show that they are complete for classes thought or known to be above **P**, such as

	flat	compressed	compressed
	short-term		long-term
	unobservable		
stationary	**PL**	**NP^{PP}**	**PSPACE**
time-dependent	**NP**	**NP^{PP}**	**NEXP**
	fully-observable		
stationary	(**P**-hard)	(**PSPACE**-hard)	(**EXP**-hard)
time-dependent	**P**	**PSPACE**	**EXP**
	partially-observable		
stationary	**NP**	**NEXP**	**NEXP**
time-dependent	**NP**	**NEXP**	**NEXP**
concise	**NP^{PP}**	**NP^{PP}**	
history-dependent	**PSPACE**	**PSPACE**	**EXPSPACE**

Table 2: The complexity of policy existence

	flat	compressed	compressed
	short horizon		long horizon
	unobservable or fully-observable		
any policy type	**NL**	**NP**	**PSPACE**
	partially-observable		
stationary	**NP**	**NP**	**NEXP**
time-dependent	**NL**	**NP**	**PSPACE**
history-dependent	**NL**	**NP**	**PSPACE**

Table 3: The complexity of policy existence with nonnegative rewards

NP, **NP^{PP}**, **PSPACE**, and even as high as **EXPSPACE**. Boutilier, Dearden, and Goldszmidt (1995) conjectured that finding optimal policies for structured POMDPs is infeasible. We prove this conjecture by showing that in many cases the complexity of our decision problems increases significantly if compressed represented POMDPs are considered, and it increases exponentially if long-term compressed represented POMDPs are considered. For example, consider short-term policy existence problems for POMDPs with nonnegative rewards are **NL**-complete under flat representations. Using compressed representations and long-term problems, the completeness increases to **PSPACE**. Compressed representations with short-term problems yield intermediate complexity results, e.g. **NP**-completeness for the above example. We observe that there is no general pattern which determines the complexity trade-offs between different restrictions. For example, we compare the complexity of policy existence problems for POMDPs with nonnegative rewards to that of POMDPs with both positive and negative rewards. Whereas for general (history-dependent) policies the complexity contrasts **NL**-completeness and **PSPACE**-completeness, for stationary policies both problems are complete for **NP**.

We prove several problems to be **NP^{PP}**-complete. In spite of its strange looking definition, **NP^{PP}** turns out to be a natural class between the Polynomial Time Hierarchy (Toda 1991) and

	optimal policy	optimal value
	is ε-approximable if and only if	
stationary	**P = NP**	**P = NP**
time-dependent	**P = NP**	**P = NP**
history-dependent		**P = PSPACE**

Table 4: Short-term non-approximability results for POMDPs with nonnegative rewards

	average performance	median performance
stationary	**PP**	**PP**
time-dependent	**PL**	**PP**
partially ordered plans	**PP**	**PP**

Table 5: The complexity of short-term average and median performances

PSPACE which deserves further investigation.

Problems that we show to be **PSPACE**- or $\mathbf{NP}^{\mathbf{PP}}$-complete are therefore computable in polynomial space. Problems in **EXP**, **NEXP**, or **EXPSPACE** are expected or known to not have polynomial-*space* algorithms, an apparently more stringent condition than not having polynomial-time algorithms. For several optimal policy resp. policy value problems (Table 4) we show that they are not polynomial-time ε-approximable unless the respective complexity class, for which the related general decision problem is complete, coincides with **P** (Section 5.4).

As we have observed, our results in some cases differ from others because we consider slightly different decision problems. Another difference between our work and many others' is the particular formalization we have chosen for representing compressed representable POMDPs.

The results for long-term policy evaluation and existence – mostly consequences from the previous sectionss – are presented in Section 7 (see also Table 1-3).

In Section 8 we consider questions that arise with *partially ordered plans*, a structure used in heuristics to find good policies for POMDPs in a restricted search space. For this, we study short-term average and median performances. Actually, we show that general hardness (and completeness) results also apply for partially ordered plans (see Table 5).

The following Section 2 gives the formal definitions of partially-observable Markov decision processes, their different specification parameters, and the related decision problems. A short overview of the complexity classes we use is given in Section 3.

For simplicity, we use abbreviations [1] for Littman, Goldsmith, and Mundhenk (1998), [2] for Mundhenk (2000), [3] for Mundhenk, Goldsmith, Lusena, and Allender (2000), and [4] for Lusena, Goldsmith, and Mundhenk (1998).

2 Partially-Observable Markov Decision Processes

A *partially-observable Markov decision process* (POMDP) describes a controlled stochastic system by its states and the consequences of actions on the system. It is denoted as a tuple $\mathcal{M} = (S, s_0, A, O, t, o, r)$, where

- S, A and O are finite sets of *states*, *actions* and *observations*,
- $s_0 \in S$ is the *initial state*,

- $t : S \times A \times S \to [0, 1]$ is the *transition probability function*, where $t(s, a, s')$ is the probability that state s' is reached from state s on action a (where $\sum_{s' \in S} t(s, a, s') \in \{0, 1\}$ for every $s \in S, a \in A$),
- $o : S \to O$ is the *observation function*, where $o(s)$ is the observation made in state s, and
- $r : S \times A \to \mathbf{Z}$ is the *reward function*, where $r(s, a)$ is the reward gained by taking action a in state s. ($\mathbf{Z}$ is the set of integers.)

If states and observations are identical, i.e. $O = S$, and if o is the identity function (or a bijection), the POMDP is called *fully-observable* and denoted as MDP. Another special case is an *unobservable* POMDP, where the set of observations contains only one element; i.e. in every state the same observation is made, and therefore the observation function is constant. We denote an unobservable POMDP as UMDP. Without restrictions on the observability, a POMDP is called *partially-observable*.[3]

2.1 Policies and Performances

Let $\mathcal{M} = (S, s_0, A, O, t, o, r)$ be a POMDP. A *step* of $\mathcal{M}$ is a transition from one state to another according to the transition probability function t. A *run* of $\mathcal{M}$ is a sequence of steps that starts in the initial state s_0. The outcome of each step is probabilistic and depends on the action chosen. A *policy* describes how to choose actions depending on observations made during the run of the process. We distinguish three types of policies (for $\mathcal{M}$).

- A *stationary policy* π_s chooses an action dependent only on the current state. This is described as a function $\pi_s : O \to A$, mapping each observation to an action.
- A *time-dependent policy* π_t chooses an action dependent only on the current state and on the number of steps the process performed already. This is described as a function $\pi_t : O \times \mathbf{N}^+ \to A$, mapping each pair $\langle$observation, time$\rangle$ to an action. ($\mathbf{N}^+$ is the set of positive integers.)
- A *history-dependent policy* π_h chooses an action dependent on all observations made during the run of the process. This is described as a function $\pi_h : O^* \to A$, mapping each finite sequence of observations to an action.

  Notice that, for an unobservable MDP, a history-dependent policy is equivalent to a time-dependent one.

A *trajectory* θ *of length* $|\theta| = m$ *for* $\mathcal{M}$ is a sequence of states $\theta = \sigma_1, \sigma_2, \ldots, \sigma_m$ ($m \geq 1$, $\sigma_i \in S$) which starts with the initial state of $\mathcal{M}$, i.e. $\sigma_1 = s_0$. Given a policy π, each trajectory θ has a probability $\text{prob}(\theta, \pi)$.

- If π is a stationary policy, then

$$\text{prob}(\theta, \pi) = \prod_{i=1}^{|\theta|-1} t(\sigma_i, \pi(o(\sigma_i)), \sigma_{i+1}) \ ,$$

[3]Note that making observations probabilistically does not add any power to POMDPs. Any probabilistically observable POMDP is equivalent to one with deterministic observations which is only polynomially larger. Thus, for the purpose of complexity analysis we can ignore this variant.

- if π is a time-dependent policy, then

$$\text{prob}(\theta, \pi) = \prod_{i=1}^{|\theta|-1} t(\sigma_i, \pi(o(\sigma_i), i), \sigma_{i+1}) \ ,$$

- if π is a history-dependent policy, then

$$\text{prob}(\theta, \pi) = \prod_{i=1}^{|\theta|-1} t(\sigma_i, \pi(o(\sigma_1) \cdots o(\sigma_i)), \sigma_{i+1}) \ .$$

We use $T_k(s)$ to denote all length k trajectories which start in the initial state s_0 and end in state s. The expected reward $R(s, k, \pi)$ obtained in state s after exactly k steps under policy π is the reward obtained in s by the action according to π weighted by the probability that s is reached after k steps.

- If π is a stationary policy, then

$$R(s, k, \pi) = r(s, \pi(o(s))) \cdot \sum_{\theta \in T_k(s)} \text{prob}(\theta, \pi) \ ,$$

- if π is a time-dependent policy, then

$$R(s, k, \pi) = r(s, \pi(o(s), k)) \cdot \sum_{\theta \in T_k(s)} \text{prob}(\theta, \pi) \ ,$$

- if π is a history-dependent policy, then the action chosen in state s depends on the observations made on the trajectory to s, and hence

$$R(s, k, \pi) = \sum_{\theta \in T_k(s), \theta = (\sigma_1, \ldots, \sigma_k)} r(s, \pi(o(\sigma_1) \cdots o(\sigma_k))) \cdot \text{prob}(\theta, \pi) \ .$$

Note that $R(s, 0, \pi) = 0$ and $R(s, 1, \pi) = r(s_0, \pi(o(s_0)))$.

A POMDP may behave differently under different policies. The quality of a policy is determined by its *performance*, i.e. by the sum of expected rewards received on it. We distinguish between different performance-metrics for POMDPs which run for a finite number of steps (also called *finite-horizon* performance), and those which run infinitely. We use $|\mathcal{M}|$ to denote the size of the representation of $\mathcal{M}$ (see Section 2.3).

- The *short-term performance of a policy* π for POMDP $\mathcal{M}$ is the expected sum of rewards received during the next $|\mathcal{M}|$ steps by following the policy π, i.e.

$$perf_s(\mathcal{M}, \pi) = \sum_{i=1}^{|\mathcal{M}|} \sum_{s \in S} R(s, i, \pi) \ .$$

- The *long-term performance of a policy* π for POMDP $\mathcal{M}$ is the expected sum of rewards received during the next $2^{|\mathcal{M}|}$ steps by following the policy π, i.e.

$$perf_l(\mathcal{M}, \pi) = \sum_{i=1}^{2^{|\mathcal{M}|}} \sum_{s \in S} R(s, i, \pi) \ .$$

- The *infinite-horizon total discounted performance* gives rewards obtained earlier in the process a higher weight than those obtained later. For $0 < \beta < 1$, the total β-discounted reward is defined as

$$perf_{td}^{\beta}(\mathcal{M}, \pi) = \sum_{i=1}^{\infty} \sum_{s \in S} \beta^i \cdot R(s, i, \pi) \; .$$

We will consider only finite-horizon performances in this work. In Section 8 we also consider the short-term *average* and *median* performances.

Let $perf_\beta$ be any performance-metric, and let α be any type of policy, either stationary, or time-dependent, or history-dependent. The α, δ-*value* $val_{\alpha,\delta}(\mathcal{M})$ of $\mathcal{M}$ is the maximal δ-performance of any policy π of type α for $\mathcal{M}$, i.e. $val_{\alpha,\delta}(\mathcal{M}) = \max_{\pi \in \mathrm{Pol}(\alpha)} perf_\delta(\mathcal{M}, \pi)$, where $\mathrm{Pol}(\alpha)$ is the set of all α policies.

2.2 Examples

We consider the example of POMDP $\mathcal{M}_{un}$ given in Figure 1. The POMDP $\mathcal{M}_{un}$ has four states

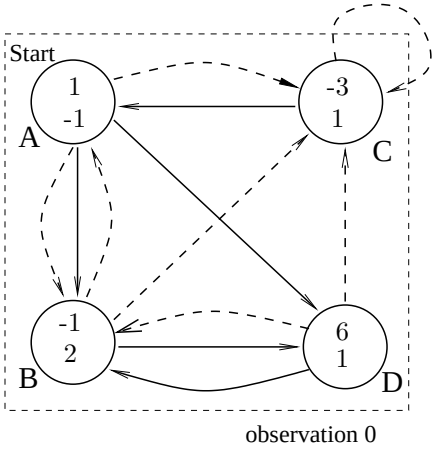


Figure 1: The example POMDP $\mathcal{M}_{un}$.

A, B, C, and D. The initial state is state A. In every state, there is a choice of two actions: either follow the dashed arcs, or follow the solid arcs. The transition probability function $t(s, \text{“dashed”}, s')$ equals 0, if there is no dashed arc from state s to s'; it equals 1, if there is a dashed arc from state s to s', and there is no other dashed arc going out from s; and it is $\frac{1}{2}$, if there is a dashed arc from state s to s', and there is another dashed arc going out from s. The reward function is described by the numbers in the circles. If in that state action “dashed” is applied, then the upper number is the reward. On action “solid”, the lower number is the reward. In all states, the same observation 0 is made. With this observation function, $\mathcal{M}_{un}$ is an unobservable POMDP.

Let us now consider the value of $\mathcal{M}_{un}$ under stationary policies for four steps. Because $\mathcal{M}_{un}$ is unobservable, there are only two possible stationary policies. The first policy is to choose the dashed arcs always, and the second policy is to choose the solid arcs always.

π_1:

observation	0
action	“dashed”

π_2:

observation	0
action	“solid”

We consider the policy π_2 with $\pi(0) =$ "solid". It yields both the trajectories with probability > 0, drawn in Figure 2. Every trajectory starts in state A (in the top row). The probability of state A

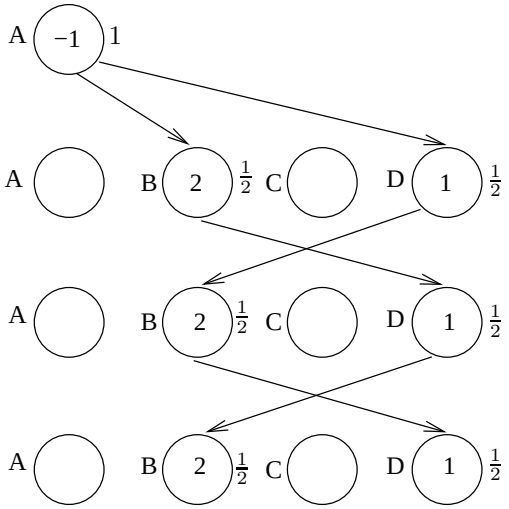


Figure 2: The trajectories of $\mathcal{M}_{un}$ under policy π_2.

being the first state equals 1 (the number to the right-hand side of the circle). The reward obtained on action "solid" in state A is -1 (the number inside the circle). Therefore, the expected reward after 1 step equals $-1 \cdot 1$. After the first step, state B or state D is reached, each with probability $\frac{1}{2}$. The reward obtained on action "solid" is 2 in state B, and it is 1 in state D. The expected reward in the second step equals $2 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2}$. In this way, we continue up to four steps and finally obtain expected reward

$$\underbrace{-1 \cdot 1}_{\text{1st step}} + \underbrace{2 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2}}_{\text{2nd step}} + \underbrace{2 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2}}_{\text{3rd step}} + \underbrace{2 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2}}_{\text{4th step}} = 3\frac{1}{2} \; .$$

This is the performance of policy π_2. Because the performance of policy π_1, that chooses the dashed arcs always, is smaller, $3\frac{1}{2}$ is the value of $\mathcal{M}_{un}$ under stationary policies (with horizon 4).

Now, let us change the observation function of $\mathcal{M}_{un}$. Say, POMDP $\mathcal{M}_{part}$ (see Figure 3) is as $\mathcal{M}_{un}$, but observation 0 is made in states A and B, and observation 1 is made in states C and D. Hence, $\mathcal{M}_{part}$ is not unobservable, but partially-observable. For $\mathcal{M}_{part}$, there are four different policies.

π_1:	observation	0	1
	action	"dashed"	"dashed"
π_2:	observation	0	1
	action	"solid"	"solid"
π_3:	observation	0	1
	action	"dashed"	"solid"
π_4:	observation	0	1
	action	"solid"	"dashed"

The policies π_1 and π_2 are essentially the same as both the policies for $\mathcal{M}_{un}$. This shows that the choice of (stationary) policies for a POMDP is at least as large as that of the respective UMDP,

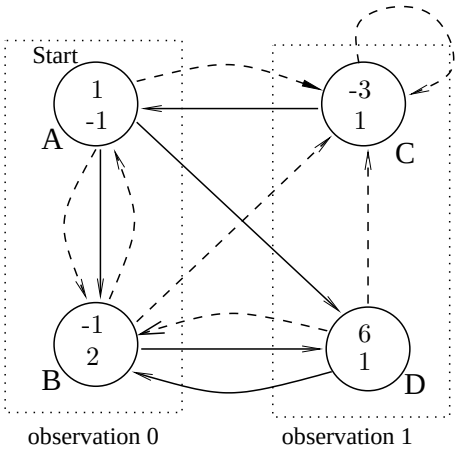


Figure 3: The POMDP $\mathcal{M}_{part}$.

and hence the value of a POMDP cannot be smaller than the value of the respective UMDP. We calculate the performance of π_4 for $\mathcal{M}_{part}$, and therefore we consider its trajectories (see Figure 4). The performance of π_4 equals

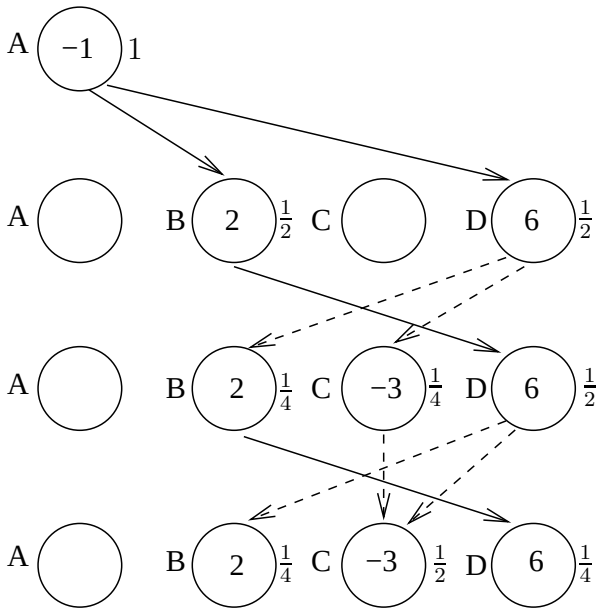


Figure 4: The trajectories of $\mathcal{M}_{part}$ under policy π_4.

$$\underbrace{-1\cdot 1}_{\text{1st step}}+\underbrace{2\cdot\frac{1}{2}+6\cdot\frac{1}{2}}_{\text{2nd step}}+\underbrace{2\cdot\frac{1}{4}-3\cdot\frac{1}{4}+6\cdot\frac{1}{2}}_{\text{3rd step}}+\underbrace{2\cdot\frac{1}{4}-3\cdot\frac{1}{2}+6\cdot\frac{1}{4}}_{\text{4th step}}=6\frac{1}{4}\ .$$

It turns out that π_3 has performance $2\frac{1}{4}$, and therefore $6\frac{1}{4}$ is the value of $\mathcal{M}_{part}$ under stationary policies (with horizon 4).

Consider the last step of the trajectories in Figure 4. The weighted rewards in state C and in state D – the states with observation 1 – sum up to 0. If action "solid" is chosen in step 4 under observation 1, we get the following time-dependent policy π'_4.

π'_4:		observation	
		0	1
	step 1	"solid"	"dashed"
	step 2	"solid"	"dashed"
	step 3	"solid"	"dashed"
	step 4	"solid"	"solid"

The performance of π'_4 calculates as that of π_4, with the only difference that the weighted rewards from state C and state D in step 4 sum up to $\frac{3}{4}$. Hence, π'_4 has performance 7. This is an example for the case that the value of a POMDP under stationary policies is worse than that under time-dependent policies. Because every stationary policy can be seen as a time-dependent policy, the time-dependent value is always at least the stationary value.

We change again the observation function and consider POMDP $\mathcal{M}_{full}$ which is as $\mathcal{M}_{part}$ or $\mathcal{M}_{un}$, but the observation function is the identity function. This means, in state A the observation is A, in state B the observation is B, and so on. $\mathcal{M}_{full}$ is a fully-observable POMDP, hence an MDP. Now we have 2^4 different stationary policies. We consider the stationary policy π_5 with

π_5:	observation	A	B	C	D
	action	"dashed"	"solid"	"solid"	"dashed"

The trajectories of $\mathcal{M}_{full}$ under π_5 can be seen in Figure 5. The performance of π_5 equals

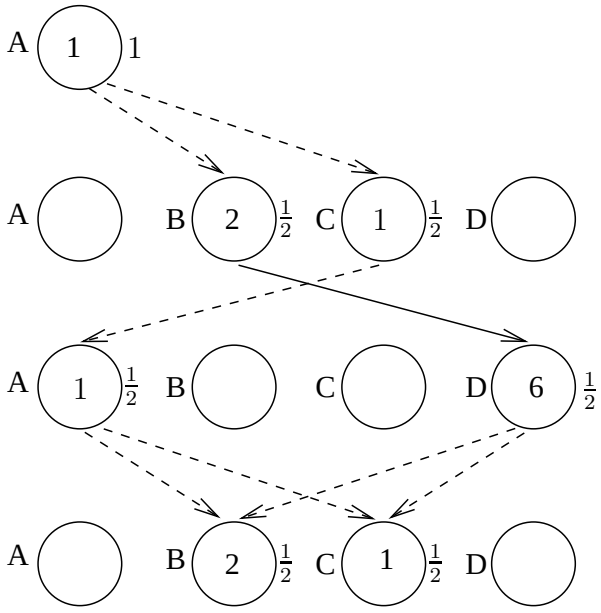


Figure 5: The trajectories of $\mathcal{M}_{full}$ under policy π_5.

$$\underbrace{1 \cdot 1}_{\text{1st step}} + \underbrace{2 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2}}_{\text{2nd step}} + \underbrace{1 \cdot \frac{1}{2} + 6 \cdot \frac{1}{2}}_{\text{3rd step}} + \underbrace{2 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2}}_{\text{4th step}} = 7\frac{1}{2} \ .$$

There is no stationary policy, under that $\mathcal{M}_{full}$ has a better performance, even though it has performance $7\frac{1}{2}$ under the following policy, too.

observation	A	B	C	D
action	"solid"	"solid"	"solid"	"dashed"

The optimal short-term policy for any MDP is a time-dependent policy. It is calculated using *backward induction*, also called *dynamic programming*. For $\mathcal{M}_{full}$, this can be done as follows. An optimal 4-step policy is calculated in 4 stages. In stage i, for each of the states s an action a is determined, such that with initial state s a policy choosing action a in the first step yields optimal reward for horizon i. In the first stage of the calculation, it is considered, which action is the best to choose in the 4th step, which is the last step. This depends only on the rewards obtained immediately in this step. The following table contains the optimal actions on the different states (=observations) in step 4, and the rewards obtained on these actions.

	A	B	C	D
step 4	"dashed" : 1	"solid" : 2	"solid" : 1	"dashed" : 6

In the next stage, we consider what to do in the previous step, i.e. in the 3rd step. For each state, one has to consider all possible actions, and to calculate the reward obtained in that state on that action plus the sum of the maximal rewards from step 4 in the above table of the states reached on that action, weighted by the probability of reaching the state. Consider e.g. state A.

1. Action "dashed" yields reward 1 in state A. On action "dashed", state B and state C are reached, each with probability $\frac{1}{2}$. From step 4, reward 2 is obtained in state B, and reward 1 is obtained in state C. Therefore, the maximal reward obtained in state A on action "dashed" sums up to $1 + \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 1 = 2\frac{1}{2}$.

2. Action "solid" yields reward -1 in state A. On action "solid", state B and state D are reached, each with probability $\frac{1}{2}$. From step 4, reward 2 is obtained in state B, and reward 6 is obtained in state C. Therefore, the maximal reward obtained in state A on action "dashed" sums up to $-1 + \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 6 = 3$.

Notice that the optimal action in state A in the 3rd step is not the action which yields the maximal immediate reward. The calculations for the other states and the other steps result in the following table.

	A	B	C	D
step 4	"dashed" : 1	"solid" : 2	"solid" : 1	"dashed" : 6
step 3	"solid" : 3	"solid" : 8	"solid" : 3	"dashed" : $7\frac{1}{2}$
step 2	"solid" : $6\frac{3}{4}$	"solid" : $9\frac{1}{2}$	"solid" : 4	"dashed" : 9
step 1	"solid" : $9\frac{1}{4}$			

This yields directly the time-dependent policy π_{opt} that achieves the maximal performance for $\mathcal{M}_{full}$. Hence, the maximal value of $\mathcal{M}_{full}$ under any policy equals $9\frac{1}{4}$. This is more than the time-dependent value of $\mathcal{M}_{part}$. But it turns out that we can find a history-dependent policy under which $\mathcal{M}_{part}$ has the same performance. Consider the trajectories of π_{opt} for $\mathcal{M}_{full}$ in Figure 6.

The history of observations of every appearance of a state in these trajectories is unique. Therefore, $\mathcal{M}_{part}$ under the following history-dependent policy has the same performance as $\mathcal{M}_{full}$ under π_{opt}.

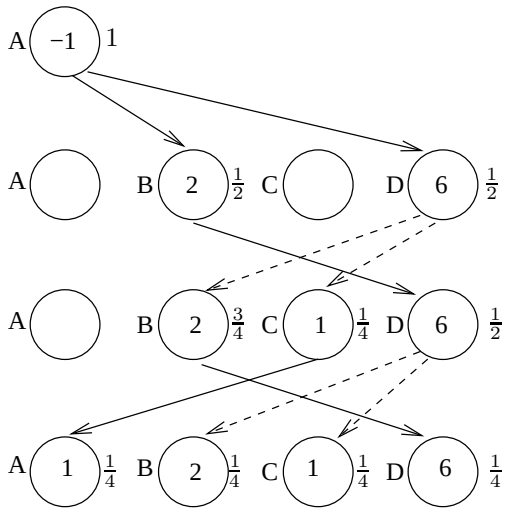


Figure 6: The trajectories of $\mathcal{M}_{full}$ under π_{opt}.

history	current observation	
	0	1
ε	"solid"	
0	"solid"	"dashed"
00		"dashed"
01	"solid"	"solid"
010	"dashed"	"dashed"
001	"solid"	"solid"

For example, on history 01 and observation 1 action "solid" is chosen. Histories which do not appear in the above trajectories are left out for simplicity.

Notice, that there is not necessarily a history-dependent policy under which a POMDP achieves the same value as the respective MDP. For example, the value of $\mathcal{M}_{un}$ under history-dependent policy is smaller than that of $\mathcal{M}_{part}$.

2.3 Representations of POMDPs

There are various ways a POMDP can be represented. The straightforward way is to write down the transition, observation, and reward function of a POMDP simply as tables. We call this the *flat representation*. When the system being modelled has sufficient structure, there is no need to represent the POMDP by complete tables. Using *compressed* and *succinct representations*, where the tables are represented by Boolean circuits, it is possible to represent a POMDP by less bits than its number of states. For example, later we will construct POMDPs that have exponentially many states compared to its representation size. Regarding the representation of a POMDP is important, since changing the representation may change the complexities of the considered decision problems too. Whereas compressed and succinct representations are commonly used by theorists, practitioners prefer e.g. *two-phase temporal Bayes nets* instead. We show how both these representations are related.

Flat representations

A *flat representation* of a POMDP with m states is a set of $m \times m$ tables for the transition function – one table for each action – and similar tables for the reward function and for the observation function. We assume that the transition probabilities are rationals that can be represented in binary as a finite string. Encodings of UMDPs and of MDPs may omit the observation function.

For a flat POMDP $\mathcal{M}$, we let $|\mathcal{M}|$ denote the number of bits used to write down $\mathcal{M}$'s tables. Note that the number of states and the number of actions of $\mathcal{M}$ is at most $|\mathcal{M}|$.

Compressed and succinct representations

When the system being modelled has sufficient structure, there is no need to represent the POMDP by complete tables. We consider circuit-based models. The first is called *succinct representations*. It was introduced independently by Galperin and Wigderson (1983), and by Wagner (1986) to model other highly structured problems. A succinct representation of a string z of length n is a Boolean circuit with $\lceil \log n \rceil$ input gates and one output gate which on input the binray number i outputs the ith bit of z. Note that any string of length n can have such a circuit of size $O(n)$, and the smallest such circuit requires $O(\log n)$ bits to simply represent i. The tables of the transition probability function for a POMDP can be represented succinctly in a similar way by a circuit which on input (s, a, s', i) outputs the ith bit of the transition probability $t(s, a, s')$. The inputs to the circuit are in binary.

The process of extracting a probability one bit at a time is less than ideal in some cases. The advantage of such a representation is that the number of bits for the represented probability can be exponentially larger than the size of the circuit. If we limit the number of bits of the probability, we can use a related representation we call *compressed*. (The issue of bit-counts in transition probabilities has arisen before; it occurs, for instance, in Beauquier, Burago, and Slissenko (1995), Tseng (1990). It is also important to note that our probabilities are specified by single bit-strings, rather than as rationals specified by two bit-strings.) We let the tables of the transition probability function for a POMDP be represented by a Boolean circuit C that on input (s, a, s') outputs all bits of $t(s, a, s')$. Note that such a circuit C has many output gates.

Encodings by circuits are no larger than the flat table encodings, but for POMDPs with sufficient structure the circuit encodings may be much smaller, namely $O(\log |S|)$, i.e., the logarithm of the size of the state space. (It takes $\lceil \log |S| \rceil$ bits to specify a state as binary input to the circuit.)

Two-phase temporal Bayes nets

There are apparently similar representations discussed in the reinforcement learning literature. Such a model is the *Two-phase temporal Bayes net* (2TBN) (Boutilier, Dearden, and Goldszmidt 1995). It describes each state of the system by a vector of values, called *fluents*. Note that if each of n fluents is two-valued, then the system has 2^n states. For simplicity (and w.l.o.g.) we assume that the fluents are two-valued and describe each fluent by a binary bit. The effects of actions are described by the effect they have on each fluent, by means of two data structures. Thus, a 2TBN be viewed as describing how the variables evolve over one step. From this, one can calculate the evolution of the variables over many steps. A 2TBN is represented as a dependency graph and a set of functions, encoded as conditional probability tables, decision trees, arithmetic decision diagrams, or in some other data structure.

Whereas for POMDPs we use a transition probability function which gives the probability $t(s, a, s')$ that state s' is reached from state s within one step on action a, a 2TBN describes a

randomized state transition function which maps state s and action a to state s' with probability $t(s, a, s')$. More abstract, a 2TBN (Boutilier, Dearden, and Goldszmidt 1995) describes a randomized function $f : \{0,1\}^m \to \{0,1\}^m$, and it is represented by a graph and a set of functions as follows.

The graph shows for each bit of the value, on which bits from the argument and from the value it depends. Each of the argument bits and each of the value bits is denoted by a node in the graph. For a function $\{0,1\}^m \to \{0,1\}^m$ we get the set of nodes $V = \{a_1, a_2, \ldots, a_m, v_1, v_2, \ldots, v_m\}$. The directed edges of the graph indicate on which bits a value bit depends. If the ith value bit depends on the jth argument bit, then there is a directed edge from a_j to v_i. Value bits may also depend on other value bits. If the ith value bit depends on the jth value bit, then there is a directed edge from v_j to v_i. This means, every edge goes from an a node to a v node, or from a v node to a v node. The graph must not contain a cycle. (Figure 7 shows an example of a 2TBN without any choice of actions.)

For each value node v_i, there is a function whose arguments are the bits on which v_i depends, and whose value is the probability that value bit i equals 1. Let f_i denote the respective function for value node v_i. Then the function f described by the 2TBN is defined as follows.

$f(x_1, x_2, \ldots, x_m) = (y_1, y_2, \ldots, y_m)$ where
$y_i = 1$ with probability $f_i(x_{j_1}, \ldots, x_{j_r}, y_{k_1}, \ldots, y_{k_l})$
for $a_{j_1}, \ldots, a_{j_r}, v_{k_1}, \ldots, v_{k_l}$ being all predecessors of v_i in the graph.

One can polynomial-time transform a probabilistic state transition function given as 2TBN into a compressed represented transition probability function (see Littman (1997b)). This means, that 2TBNs are not a more compact representation than the compressed representation. It is not so clear how to transform a compressed represented transition probability function into a 2TBN. This has nothing to do with the representation, but with a general difference between a probabilistic function and a deterministic function that computes a probability. Assume, that for a transition probability function t we have a circuit C that calculates the probability distribution of t,

$$t_\Sigma(s, a, s') = \sum_{q \le s'} t(s, a, q) \ .$$

Here, we assume to have the states ordered, e.g. in lexicographic order. The circuit C can be transformed into a *randomized Boolean circuit* that calculates the according probabilistic state transition T where $T(s, a) = s'$ with probability $t(s, a, s')$. A randomized circuit consists of input gates, AND, OR, and NOT gates (which calculate the logical conjunction, disjunction, resp. negation of its input(s)), and of "random gates", which have no input and output either 0 or 1 both with probability $\frac{1}{2}$. On input (s, a) and random bits r, the randomized circuit uses binary search to calculate the state s' such that $t_\Sigma(s, a, s' - 1) < r \le t_\Sigma(s, a, s')$, where $s' - 1$ is the predecessor of s' in the order of states. Finally, this state s' is the output. Given circuit C, this transformation can be performed in polynomial time. Notice that Sutton and Barto (1998) used randomized circuits to represent probabilistic state transition functions.

We will now show how to transform a randomized Boolean circuit into a 2TBN, which describes essentially the same function. Let R be a randomized Boolean circuit which describes a function $f_R : \{0,1\}^n \to \{0,1\}^{n'}$, and let T be a 2TBN, which describes a function $f_T : \{0,1\}^m \to \{0,1\}^m$ where $m \ge \max(n, n')$. We say that T *simulates* R if on the common variables both calculate the same, independent on how the other input variables are set. Formally, T simulates R if for all $i_1, \ldots, i_m \in \{0,1\}$ the following holds:

$$\begin{aligned} &\mathrm{Prob}[f_R(i_1, \ldots, i_n) = (o_1, \ldots, o_{n'})] \ = \\ &\qquad \sum_{o_{n'+1}, \ldots, o_m} \mathrm{Prob}[f_T(i_1, \ldots, i_m) = (o_1, \ldots, o_{n'}, o_{n'+1}, \ldots, o_m)]. \end{aligned}$$

Lemma 2.1 *There is a polynomial-time computable function which on input a randomized Boolean circuit R outputs a 2TBN which simulates R.*

Proof. Let R be a circuit with n input gates and n' output gates. The outcome of the circuit on any input $b_1, \ldots, b_n$ is usually calculated as follows. At first, calculate the outcome of all gates, which depend only on input gates. Next, calculate the outcome of all gates, which depend only on those gates, whose outcome is already calculated, and so on. This yields an enumeration of the gates of a circuit, such that the outcome of a gate can be calculated when all the outcomes of gates with a smaller index are already calculated. We assume that the gates are enumerated in this way, that $g_1, \ldots, g_n$ are the input gates, and that $g_l, \ldots, g_s$ are the other gates, where the smallest index of a gate which is neither an output nor an input gate equals $l = \max(n, n') + 1$.

Now, we define a 2TBN as follows. For every index $i \in \{1, 2, \ldots, s\}$ we take two nodes, one for the ith argument bit and one for the ith value bit. This yields a set of nodes $a_1, \ldots, a_s, v_1, \ldots, v_s$. If an input gate g_i $(1 \leq i \leq n)$ is input to gate g_j, then we get an edge from a_i to v_j. If the output of a non input gate g_i $(n < i < m)$ is input to gate g_j, then we get an edge from v_i to v_j. Finally, the nodes $v_1, \ldots, v_o$ stand for the value bits. If gate g_j produces the ith output bit, then there is an edge from v_j to v_i. Because the circuit C has no loop, the graph is loop free, too.

The functions associated to the nodes $v_1, \ldots, v_m$ depend on the functions calculated by the respective gate and are as follows. Every of the value nodes v_i $(i = 1, 2, \ldots, n')$ has exactly one predecessor, whose value is copied into v_i. Hence, f_i is the one-place identity function, $f_i(x) = x$ with probability 1. Now we consider the nodes which come from internal gates of the circuit. If g_i is an AND gate, then $f_i(x, y) = x \wedge y$, if g_i is an OR gate, then $f_i(x, y) = x \vee y$, and if g_i is a NOT gate, then $f_i(x) = \neg x$, all with probability 1. If g_i is a random gate, then f_i is a function without argument where $f_i = 1$ with probability $\frac{1}{2}$.

By this construction, it follows that the 2TBN T simulates the randomized Boolean circuit R.

An example of a randomized circuit and the 2TBN to which it is transformed as described above, is given in Figure 7. □

Concluded, we can say that compressed representations and representations as 2TBNs are very similar.

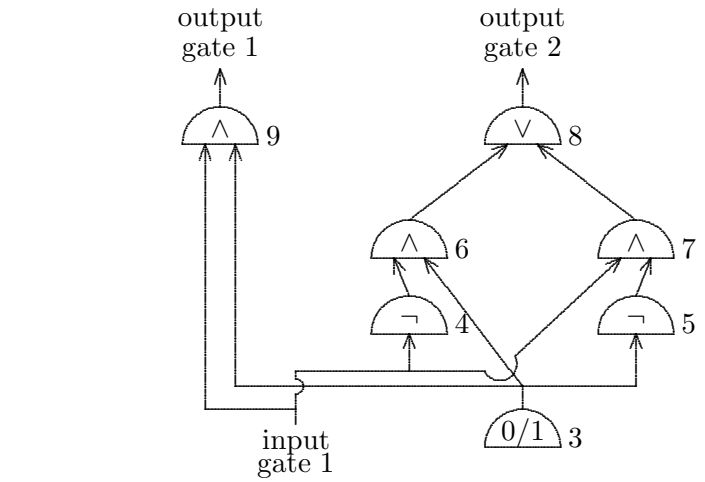
Theorem 2.2 *1. There is a polynomial-time computable function, that on input a POMDP whose state transition function is represented as a 2TBN, outputs a compressed representation of the same POMDP.*

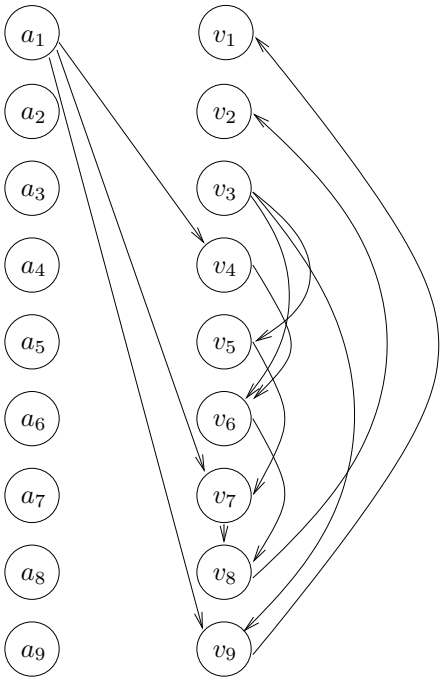
2. There is a polynomial-time computable function, that on input a POMDP whose probability distribution function t_Σ is compressed represented, outputs the same POMDP as a 2TBN.

2.4 Representations of Policies

For a flat represented POMDP $\mathcal{M}$, a stationary policy π can be encoded as a list with at most $|\mathcal{M}|$ entries, each entry of at most $|\mathcal{M}|$ bits. A short-term time-dependent policy can be encoded as a table with at most $|\mathcal{M}| \cdot |\mathcal{M}|$ entries each of at most $|\mathcal{M}|$ bits. This means that for a POMDP $\mathcal{M}$, any stationary policy and any short-term time-dependent policy can be specified with polynomially in $|\mathcal{M}|$ many bits – namely at most $|\mathcal{M}|^3$ many.

This is not the case for short-term history-dependent policies. There are $\sum_{i=0}^{|\mathcal{M}|-1} |O|^i$ many different histories of up to $|\mathcal{M}|$ steps with observations O. If there at least 2 observations in O, this sum is exponential in $|\mathcal{M}|$. The representation of a history-dependent policy can therefore be exponentially large in the size of the POMDP. We call this the *intractability of representing a*





f_1 :

v_8	$\text{Prob}(v_1 = 1)$
0	0
1	1

f_3 :

$\text{Prob}(v_3 = 1)$
$\frac{1}{2}$

$\vdots$

f_9 :

a_1	v_3	$\text{Prob}(v_9 = 1)$
0	0	0
0	1	0
1	0	0
1	1	1

Figure 7: A randomized Boolean circuit which outputs the binary sum of its input and a random bit (gate 3), and a 2TBN representing the circuit (only functions f_1, f_3 and f_9 are described)

history-dependent policy. Similar as for the tractable representation of a POMDP with huge state space, we alternatively can take circuits to represent history-dependent policies more compact. Consider a policy π for horizon k for a POMDP with m observations and a actions. A circuit representing this policy has $k \cdot \lceil \log m + 1 \rceil$ input gates and $\lceil \log a \rceil$ output gates. The input gates are considered to be split into blocks of length $\lceil \log m + 1 \rceil$, whose ith block is used to input the ith observation or the "empty" observation in case that an action for a step smaller than i should be calculated. On input a sequence of observations, the action chosen by the policy can be read from the output gates of the circuit. A history-dependent policy represented as circuit is called *concise*. For any constant $c > 0$, we say that a policy π for $\mathcal{M}$ is *c-concise*, if π is a history-dependent policy for horizon $|\mathcal{M}|$ and can be represented by a circuit C of size $|\mathcal{M}|^c$. Notice that every time-dependent policy for a flat POMDP is concise. A circuit representing a concise policy for a long-term run of a POMDP has exponentially many inputs. Hence, it is not tractable to represent such a policy.

2.5 Computational Questions

Now we are ready to define the problems whose complexity we will investigate in the rest of this work. The standard computational problems associated with POMDPs are

- the calculation of the performance of a policy for a POMDP,
- the calculation of the value of a POMDP, and
- the calculation of an optimal policy for a POMDP.

For each POMDP, the following parameters influence the complexity of the calculations.

- Representation (flat, compressed),
- rewards (nonnegative, both negative and nonnegative)
- type of policy
  (stationary, time-dependent, concise, history-dependent),
- and performance-metric (short-term, long-term).

We see representation and rewards as part of the POMDP. This yields the following problems.

Performance problem:

given a POMDP, a performance-metric, and a policy

calculate the performance of the given policy under the given performance-metric.

Value problem:

given a POMDP, a performance-metric, and a policy type,

calculate the value of the POMDP under the specified type of policy and the given performance-metric.

Optimal policy problem:

given a POMDP, a performance-metric, and a policy type,

calculate a policy whose performance is the respective value of the POMDP.

All these are *functional* problems. However, standard complexity results consider *decision problems*. Papadimitriou and Tsitsiklis (1987) considered the question for POMDPs with nonpositive rewards, of whether there was a policy with performance 0. Our POMDPs have both negative and nonnegative rewards, what makes the following decision problems more general.

For each type of POMDP in any representation, each type of policy, and each performance-metric, we consider the *policy evaluation problem*, and the *policy existence problem*.

Policy evaluation problem:

given a POMDP, a performance-metric, and a policy,

decide whether the performance of the POMDP under that policy and that performance-metric is greater 0.

Policy existence problem:

given a POMDP, a performance-metric, and a policy type,

decide whether the value of the POMDP under the specified type of policy and the given performance-metric is greater 0.

Finite-horizon problems with other performance-metrices than short-term or long-term often reduce to the problems established here. For example, consider the problem given a POMDP $\mathcal{M}$, a polynomially bound and polynomial-time computable function f, and a policy π, calculate the performance of $\mathcal{M}$ under π after $f(|\mathcal{M}|)$ steps. We construct a new POMDP $\mathcal{M}'$ that consists of $f(|\mathcal{M}|)$ copies of $\mathcal{M}$. The initial state of $\mathcal{M}'$ is the initial state of the first copy. If there is a transition from s to s' in $\mathcal{M}$, then in $\mathcal{M}$ there is a transition from s in the ith copy to s' in the $i+1$st copy. The observation from a state s is the same in all copies, and also the rewards are the same. All transitions from the last copy go to a sink state, in which the process remains on every action without rewards. Because f is polynomially bound, $\mathcal{M}'$ can be constructed in polynomial time. It follows straightforwardly that the performance of $\mathcal{M}'$ after $|\mathcal{M}'|$ steps is the same as the performance of $\mathcal{M}$ after $f(|\mathcal{M}|)$ steps. Hence, the problem reduces to the performance problem. Similar reductions work for the other problems, for example for the finite-horizon total discounted performance. Notice that for the policy existence problem, partially-observability is necessary to make this reduction work. This means, it transforms a fully-observable POMDP into a POMDP that is not fully-observable. Hence, this reduction does not work for MDPs.

Using a binary search technique and the policy existence problem as an "oracle", it is possible to compute the value problem and the optimal policy problem for short-term stationary and time-dependent policies in polynomial time (relative to the oracle). In the same way, the performance problem can be calculated in polynomial time relative to the policy evaluation problem. In this sense, the functional problems and the decision problems are polynomially equivalent.

3 Complexity Classes

For definitions of complexity classes, reductions, and standard results from complexity theory we refer to the textbook of Papadimitriou (1994). In the interest of completeness, in this section we give a short description of the complexity classes and their complete decision problems used later in this work. These problems have to do with graphs, circuits, and formulas. A *graph* consists of a set of nodes and a set of edges between nodes. A typical graph problem is the *graph reachability problem*: given a graph, a source node, and a sink node, decide whether the graph has a path from the source to the sink. A *circuit* is a cycle-free graph, whose vertices are labeled with gate types AND, OR, NOT, INPUT. The nodes labeled INPUT have indegree 0, i.e. there are no edges that end in an INPUT node. There is only one node with outdegree 0, and this node is called the output gate. (See Figure 10 for an example of a circuit.) A typical circuit problem is the *circuit value problem* CVP: given a circuit and an input, decide whether the circuit outputs 1. A *formula* consists of variables x_i and constants (0 or 1), operators $\wedge$ ("and", conjunction), $\vee$ ("or", disjunction), and $\neg$ ("not", negation), and balanced parentheses. For example, $(x_1 \wedge \neg x_2) \vee \neg(x_2 \vee \neg x_1)$ is a formula. A formula is in *conjunctive normal form* (CNF) if it is a conjunction of disjunctions of literals, i.e. variables and negated variables. It is in 3CNF, if every disjunction has at most 3 literals. For example, the formula $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_1)$ is in 3CNF. If ϕ is the formula $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_1)$, then we write $\phi(b_1, b_2, b_3)$ to denote the formula obtained from ϕ by replacing each variable x_i by $b_i \in \{0, 1\}$. E.g. $\phi(1, 0, 1) = (1 \vee \neg 0 \vee 1) \wedge (0 \vee \neg 1)$, and $\phi(1, 0) = (1 \vee \neg 0 \vee x_3) \wedge (0 \vee \neg 1)$. Notice that for a formula ϕ with n variables, $\phi(b_1, \ldots, b_n)$ is either true or false. Instead of true and false we also use the values 1 and 0. The *satisfiability problem* for formulas is: given a formula ϕ, decide whether $\phi(b_1, \ldots, b_n)$ is true for some choice of $b_1, \ldots, b_n \in \{0, 1\}$. The problem 3SAT is the same, but only formulas in 3CNF are considered. Formulas can also be quantified. Quantified formulas are either true or false. E.g., the formula $\exists x_1 \forall x_2 (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ is false. The problem QBF asks given a quantified formula, to decide whether it is true.

It is important to note that most of the complexity classes we consider are *decision classes.* This means that the problems in these classes are questions with "yes/no" answers. Thus, for instance, the traveling salesperson problem is in **NP** in the form, "Is there a TSP tour within budget b?" The question of finding the *best* TSP tour for a graph is technically not in **NP**, although the decision problem can be used to find the optimal value via binary search. Although there is an optimization class associated with **NP**, there are not common optimization classes associated with all of the decision classes we reference. Therefore, we have phrased our problems as decision problems in order to use known complete problems for these classes.

The sets decidable in polynomial time form the class **P**. Decision problems in **P** are often said to be "tractable," or more commonly, problems that cannot be solved in polynomial time are said to be intractable. A standard complete problem for this class is the circuit value problem (CVP).

The class **EXP** consists of sets decidable in exponential time. This is the smallest class considered in this work which is known to contain **P** properly. The complete set we use for **EXP** is the *succinct circuit value problem* (sCVP). This is a similar problem to CVP, except that the circuit is represented succinctly. I.e. it is given in terms of two (possibly much smaller) circuits, one that takes as input i and j and outputs a 1 if and only if i is the parent of j, and a second circuit that takes as input i and a gate label t, and outputs 1 if and only if i is of labeled t.

The nondeterministic variants of **P** and **EXP** are **NP** and **NEXP**. The complete sets used here are primarily the satisfiability problem (proven in Cook's Theorem), 3SAT and succinct 3SAT (the formula has at most 3 appearances of every variable and is again given by a circuit).

The sets decidable in polynomial space form the class **PSPACE**. The most common **PSPACE**-

complete problem is QBF. The class **L** is the class of sets which are decidable by logarithmic-space bounded Turing machines. Such machines have limits on the space used as "scratch space," and have a read-only input tape. The nondeterministic variants of **L** and **PSPACE** are **NL** and **NPSPACE**. A complete problem for **NL** is the graph reachability problem. The class **NPSPACE** is equal to **PSPACE**, i.e. with polynomial-space bounded computation nondeterminism does not help. The class **EXPSPACE** consists of sets decidable in exponential space.

Nondeterministic computation is essential for the definitions of probabilistic and counting complexity classes. The class #**L** (Àlvarez and Jenner 1993) is the class of functions f such that, for some nondeterministic logarithmic-space bounded machine N, the number of accepting paths of N on x equals $f(x)$. The class #**P** is defined analogously as the class of functions f such that, for some nondeterministic *polynomial-time* bounded machine N, the number of accepting paths of N on x equals $f(x)$.

Probabilistic logarithmic-space, **PL** , is the class of sets A for which there exists a nondeterministic logarithmically space-bounded machine N such that $x \in A$ if and only if the number of accepting paths of N on x is greater than its number of rejecting paths. In apparent contrast to **P**-complete sets, sets in **PL** are decidable using very fast parallel computations (Jung 1985). Probabilistic polynomial time, **PP** , is defined analogously. A classic **PP**-complete problem is majority satisfiability (MAJSAT): given a Boolean formula, does the majority of assignments satisfy it?

For polynomial-space bounded computations, **PSPACE** equals probabilistic **PSPACE**, and #**PSPACE** (defined analogously to #**L** and #**P**) is the same as the class of polynomial-space computable functions (Ladner 1989). Note that functions in #**PSPACE** produce output up to exponential length.

Another interesting complexity class is $\mathbf{NP}^{\mathbf{PP}}$. As for each member of an **NP** set there is a short certificate of its membership which can be checked in polynomial time, for each member of an $\mathbf{NP}^{\mathbf{PP}}$ set there is a short certificate of its membership which can be checked in probabilistic polynomial time (Torán 1991). A typical problem for $\mathbf{NP}^{\mathbf{PP}}$ is EMAJSAT: given a pair (ϕ, k) consisting of a Boolean formula ϕ of n variables $x_1, \ldots, x_n$ and a number $1 \leq k \leq n$, is there an assignment to the first k variables $x_1, \ldots, x_k$ such that the majority of assignments to the remaining $n-k$ variables $x_{k+1}, \ldots, x_n$ satisfies ϕ? I.e. are there $b_1, \ldots, b_k \in \{0,1\}$ such that $\phi(b_1, \ldots, b_k) \in$ MAJSAT?

For $k = n$, this is precisely the satisfiability problem, the classic **NP**-complete problem. This is because we are asking whether there exists an assignment to *all* the variables that makes ϕ true. For $k = 0$, EMAJSAT is precisely MAJSAT, the **PP**-complete problem. This is because we are asking whether the majority of all *total* assignments makes ϕ true.

Deciding an instance of EMAJSAT for intermediate values of k has a different character. It involves both an **NP**-type calculation to pick a good setting for the first k variables and a **PP**-type calculation to see if the majority of assignments to the remaining variables makes ϕ true. This is akin to searching for a good answer (plan, schedule, coloring, belief network explanation, etc.) in a combinatorial space when "good" is determined by a computation over probabilistic quantities. This is just the type of computation described by the class $\mathbf{NP}^{\mathbf{PP}}$.

Theorem 3.1 [1] EMAJSAT *is* $\mathbf{NP}^{\mathbf{PP}}$*-complete.*

Proof sketch. Membership in $\mathbf{NP}^{\mathbf{PP}}$follows directly from definitions. To show completeness of EMAJSAT, we first observe (Torán 1991) that every set in $\mathbf{NP}^{\mathbf{PP}}$can be $\leq_m^{\mathbf{NP}}$ to the **PP**-complete set MAJSAT. Thus, any $\mathbf{NP}^{\mathbf{PP}}$computation can be modeled by a nondeterministic machine N that, on each possible computation, first guesses a sequence s of bits that controls its nondeterministic moves, deterministically performs some computation on input x and s, and then writes down a

formula $q_{x,s}$ with variables in $z_1, \ldots, z_l$ as a query to MAJSAT. Finally, $N(x)$ with oracle MAJSAT accepts if and only if for some s, $q_{x,s} \in$ MAJSAT.

Given any input x, like in the proof of Cook's Theorem, we can construct a formula ϕ_x with variables $y_1, \ldots, y_k$ and $z_1, \ldots, z_l$ such that for every assignment $a_1, \ldots, a_k, b_1, \ldots, b_l$ it holds that

$$\phi_x(a_1, \ldots, a_k, b_1, \ldots, b_l) = q_{x,a_1 \cdots a_k}(b_1, \ldots, b_l).$$

Thus, $(\phi_x, k) \in$ EMAJSAT if and only if for some assignment s to $y_1, \ldots, y_k$, $q_{x,s} \in$ MAJSAT if and only if $N(x)$ accepts. ■

For the complexity classes mentioned, the following inclusions hold.

- $\mathbf{NL} \subseteq \mathbf{PL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PP} \subseteq \mathbf{NP}^{\mathbf{PP}} \subseteq \mathbf{PSPACE}$
- $\mathbf{PSPACE} \subseteq \mathbf{EXP} \subseteq \mathbf{NEXP} \subseteq \mathbf{EXPSPACE}$
- $\mathbf{NL} \subseteq \mathbf{PL} \subset \mathbf{PSPACE} = \mathbf{NPSPACE} \subset \mathbf{EXPSPACE}$
- $\mathbf{P} \subset \mathbf{EXP}$ and $\mathbf{NP} \subset \mathbf{NEXP}$

4 Short-Term Policy Evaluation

The policy evaluation problem asks whether a given POMDP $\mathcal{M}$ has performance greater than 0 under a given policy π after $|\mathcal{M}|$ steps. For the policy evaluation problem, the observability of the POMDP does not matter. Therefore, we state the results mainly for general partially-observable Markov decision processes.

We consider the complexity of the policy evaluation problems for flat POMDPs in Section 4.1. An instance of this problem consists of a POMDP and a policy. Hence, the complexity is quantified in the length of both together. For history-dependent policies, because of the intractability of their representation, the policy-evaluation problem gets an unusually low complexity. Short-term stationary, time-dependent and c-concise policies do not cause such problems. Concise (history-dependent) policies are the only that may have representations of size polynomial in the size of the POMDP representation for flat and compressed POMDPs. In Section 4.2 evaluation problems for compressed POMDPs are considered. General history-dependent policies have size double exponential in that of the respective compressed represented POMDP. Because this again yields a "pathologically" low complexity of the evaluation problem, we do not consider it in this context.

4.1 Flat Representations

The long known standard polynomial-time algorithm for evaluating a given policy for a given POMDP uses dynamic programming (see for example Papadimitriou and Tsitsiklis (1987), Puterman (1994)). We give a major improvement by showing that for stationary and time-dependent policies this evaluation can be performed quickly in parallel. Eventually this yields these policy evaluation problems being complete for **PL**. For concise policies, a different approach yields **PP**-completeness.

Stationary and time-dependent policies

We begin with a technical lemma about matrix powering, and show that each entry $(T^m)_{(i,j)}$ of the mth power of a nonnegative integer square matrix T can be computed in $\#\mathbf{L}$, if the power m is at most the dimension n of the matrix.

Lemma 4.1 *(cf. Vinay (1991)) Let T be an $n \times n$ matrix of nonnegative binary integers, each of length n, and let $1 \leq i, j \leq n$, $0 \leq m \leq n$. The function mapping (T, m, i, j) to $(T^m)_{(i,j)}$ is in $\#\mathbf{L}$.*

We review the relations between the function class $\#\mathbf{L}$ and the class of decision problems **PL**. **GapL** is the class of functions representable as differences of $\#\mathbf{L}$ functions, $\mathbf{GapL} = \{g - h \mid g, h \in \#\mathbf{L}\}$. The class **PL** consists of sets A for which there is a **GapL** function f such that for every x, $x \in A$ if and only if $f(x) > 0$ (see Allender and Ogihara (1996)). We use these results to prove the following.

Lemma 4.2 [3] *The stationary policy evaluation problem for POMDPs is in* **PL**.

Proof. Let $\widehat{\mathcal{M}} = (S, s_0, A, O, \hat{t}, o, \hat{r})$ be a POMDP, and let π be a stationary policy, i.e. a mapping from O to A. We show how $\mathit{perf}_s(\widehat{\mathcal{M}}, \pi) > 0$ can be decided in **PL**.

We transform $\widehat{\mathcal{M}}$ into a slightly simpler POMDP $\mathcal{M}$ with the same states and rewards as $\widehat{\mathcal{M}}$, having the same performance as $\widehat{\mathcal{M}}$ under policy π. We obtain $\mathcal{M}$ by renaming the actions in a way, that π chooses the same action – say action a – under all observations. The state transition function $\hat{t}$ and the reward function $\hat{r}$ are changed accordingly. Let $\mathcal{M} = (S, s_0, A, O, t, o, r)$ be the POMDP obtained by this renaming. Notice that the renaming can be performed in logarithmic space, and it does not change the size of $\widehat{\mathcal{M}}$, i.e. $|\widehat{\mathcal{M}}| = |\mathcal{M}|$. Then $\widehat{\mathcal{M}}$ under policy π has the same performance as $\mathcal{M}$ under (constant) policy a, i.e. $\mathit{perf}_s(\widehat{\mathcal{M}}, \pi) = \mathit{perf}_s(\mathcal{M}, a)$. This performance can be calculated using an inductive approach. Let $\mathit{perf}(\mathcal{M}, s, m, a)$ be the performance of $\mathcal{M}$ under policy a when started in state s for m steps. By the definition of perf_s, it holds that $\mathit{perf}_s(\mathcal{M}, a) = \mathit{perf}(\mathcal{M}, s_0, |\mathcal{M}|, a)$. It is not hard to see that perf can be inductively defined as follows.

- $\mathit{perf}(\mathcal{M}, s, 0, a) = 0$ for every state $s \in S$, and
- $\mathit{perf}(\mathcal{M}, s, m, a) = r(s, a) + \sum_{j \in S} t(s, a, j) \cdot \mathit{perf}(\mathcal{M}, j, m-1, a)$, for every state $s \in S$ and every $m \geq 1$.

The performance may be a rational number. Because the complexity class **GapL** deals with integers, we need to make the performance an integer. The state transition probabilities are given as binary fractions of length h. In order to get an integer function for the performance, define the function p as $p(\mathcal{M}, i, 0) = 0$ and $p(\mathcal{M}, i, m) = 2^{hm} \cdot r(i, a) + \sum_{j \in S} p(\mathcal{M}, j, m-1) \cdot 2^h \cdot t(i, a, j)$. One can show that

$$\mathit{perf}(\mathcal{M}, i, m, a) = p(\mathcal{M}, i, m) \cdot 2^{-hm} \quad .$$

Therefore, $\mathit{perf}(\mathcal{M}, s_0, |\mathcal{M}|, a) > 0$ if and only if $p(\mathcal{M}, s_0, |\mathcal{M}|) > 0$.

In order to complete the proof using the characterization of **PL** mentioned above, we have to show that the function p is in **GapL**.

Let T be the matrix obtained from the transition probability function for action a of $\mathcal{M}$ by multiplying all entries by 2^h, i.e. $T_{(i,j)} = t(i, a, j) \cdot 2^h$. The recursion in the definition of p can be resolved into powers of T, and we get

$$p(\mathcal{M}, i, m) = \sum_{k=1}^{m} \sum_{j \in S} (T^{k-1})_{(i,j)} \cdot r(j, a) \cdot 2^{(m-k+1) \cdot h} \quad .$$

Each $T_{(i,j)}$ is computable in logarithmic space from the input $\widehat{\mathcal{M}}$. From Lemma 4.1 we get that $(T^{k-1})_{(i,j)} \in \#\mathbf{L}$. The reward function is part of the input too, thus r is in **GapL** (note that

rewards may be negative integers). Because **GapL** is closed under multiplication and polynomial summation (see Allender and Ogihara (1996)), it follows that $p \in \mathbf{GapL}$. □

In the following hardness proof, we simulate a Turing machine computation by a POMDP which has only one action.

Lemma 4.3 [3] *The stationary policy evaluation problem for POMDPs is* **PL***-hard.*

Proof. Consider $B \in \mathbf{PL}$. Then there exists a probabilistic logspace machine N accepting B, and a polynomial p such that each computation of N on x uses at most $p(|x|)$ random decisions (Jung 1985). Now, fix some input x. We construct a UMDP $\mathcal{M}(x)$ with only one action a, which models the behavior of N on x. Each state of $\mathcal{M}(x)$ is a pair consisting of a configuration of N on x (there are polynomially many) and an integer used as a counter for the number of random moves made to reach this configuration (there are at most $p(|x|)$ many). Also, we add a final "trap" state reached from states containing a halting configuration or from itself. The transition probability function of $\mathcal{M}(x)$ is defined according to the transition probability function of N on x, so that each halting computation of N on x corresponds to a length $p(|x|)$ trajectory of $\mathcal{M}(x)$ and vice versa. The transition probability function is defined so that each halting computation of N on x corresponds to a length $p(|x|)$ trajectory of $\mathcal{M}(x)$ and vice versa. The reward function is chosen such that trajectories θ corresponding to accepting computations yield reward $\frac{1}{\text{prob}(\theta,a)}$ in their final step $\frac{1}{\text{prob}(\theta,a)}$, and trajectories θ corresponding to rejecting computations yield reward $-\frac{1}{\text{prob}(\theta,a)}$. The reward can be determined by the number of random steps made, as recorded in the counter. If we weight the reward of a trajectory with its probability, we get 1 for each "accepting" trajectory and -1 for each "rejecting" trajectory. Trajectories which neither reach an accepting state nor an rejecting state yield reward 0. Since $x \in B$ if and only if the number of accepting computations of N on x is greater than the number of rejecting computations, it follows that $x \in B$ if and only if the number of trajectories θ of length $|\mathcal{M}(x)|$ for $\mathcal{M}(x)$ with weighted reward 1 is greater than the number of trajectories with weighted reward -1, which is equivalent to $perf_s(\mathcal{M}(x), a) > 0$. □

Clearly, if there is no choice of actions, there is only one policy: take this action. Therefore, this hardness proof also applies for unobservable and fully-observable POMDPs.

Corollary 4.4 [3] *The stationary policy evaluation problem for UMDPs and for MDPs is* **PL***-hard.*

PL-completeness of the stationary policy evaluation problem follows immediately from the above Lemmas 4.2 and 4.3. To get the same result for time-dependent policies, it takes only a slightly modified probabilistic logspace algorithm.

Theorem 4.5 [3] *The stationary and time-dependent policy evaluation problems for POMDPs are* **PL***-complete.*

Proof. We consider the time-dependent case. In the proof of Lemma 4.3, a POMDP with only one action was constructed. Hence, instead of transforming a **PL** computation into a POMDP and a stationary policy, that always takes this action, one can transform a **PL** computation into a POMDP and a time-dependent policy, that always takes this action. So, hardness of the time-dependent case follows from Lemma 4.3. It remains to show that the time-dependent case is contained in **PL**. Let $\widehat{\mathcal{M}} = (S, s_0, A, O, \hat{t}, \hat{o}, \hat{r})$ be a POMDP, and let π be a time-dependent policy, i.e. a mapping from $O \times \{1, 2, \ldots, |\widehat{\mathcal{M}}|\}$ to A. Essentially, we proceed in the same way as in the above proof. The main difference is that the transformation from $\widehat{\mathcal{M}}$ to $\mathcal{M}$ is more involved. We construct $\mathcal{M}$ by

making $|\widehat{\mathcal{M}}|$ copies of $\widehat{\mathcal{M}}$ such that all transitions from the ith copy go to the $i+1$st copy, and all these transitions correspond to the transition chosen by π in the ith step. The rest of the proof proceeds as in the proof of Lemma 4.2. □

As already argued above, the same techniques can be applied for UMDPs and MDPs.

Corollary 4.6 [3] *The stationary and time-dependent policy evaluation problems for UMDPs and for MDPs are* **PL***-complete.*

Papadimitriou and Tsitsiklis (1987) considered POMDPs with *nonpositive* rewards. We consider the policy evaluation problems – and later the existence problems – for POMDPs with *nonnegative* rewards, which fits better to our question for a *positive* performance. This is generally easier than for POMDPs with unrestricted rewards. It suffices to find *one* trajectory with positive probability through the given POMDP that is consistent with the given policy and yields reward > 0 in at least one step. Moreover, the transition probabilities and rewards do not need to be calculated exactly. In fact, these problems reduce in a straightforward way to graph accessibility problems. Thus, our question has a "there exists" flavor, whereas the question by Papadimitriou and Tsitsiklis (1987) mentioned above has a "for all" flavor.

Theorem 4.7 [3] *The stationary and time-dependent policy evaluation problems for POMDPs with nonnegative rewards are* **NL***-complete.*

Proof. Neglecting rewards and exact transition probabilities, a POMDP $\mathcal{M}$ (any observability) and a stationary policy π can be interpreted as a directed graph: the vertices are states of the POMDP, and an edge exists from s_i to s_j if and only if $t(s_i, \pi(o(s_i)), s_j) > 0$. Then $perf_s(\mathcal{M}, \pi) > 0$ if and only if a vertex of the set $G = \{s_i : \ r(s_i, \pi(o(s_i))) > 0\}$ is reachable from s_0. This is an instance of the standard **NL**-complete graph reachability problem. Because a path from s_0 to a vertex in G contains any node at most once, the same idea holds for time-dependent policies.

The **NL**-complete graph reachability problem can easily be transformed to a performance problem for a POMDP. Any graph is transformed to a POMDP with only *one* action (all edges from a given vertex have equal probability). The source of the graph is the initial state of the POMDP, and the sink of the graph is the only state of the POMDP whose incoming edges yield reward 1 – all other rewards are 0. The POMDP has only one policy, and that policy has positive performance if and only if the sink node is reachable in the original graph. Since there is only one possible policy, observability is irrelevant. □

Concise and history-dependent policies

Concise policies are a restriction of history-dependent policies and more general than time-dependent policies. Other than history-dependent policies in general, they may be representable in size polynomial in the length of the horizon. Concise policies seem to be the most general polynomially representable policy type. Its policy evaluation problem is much harder than for the other types of policies.

Lemma 4.8 [2] *The concise policy evaluation problem for POMDPs is in* **PP***.*

Proof. We show how to evaluate a process $\mathcal{M}$ under a concise policy π given as a circuit. Let $\mathcal{M} = (S, s_0, A, O, t, o, r)$ and π be given. Let h be the maximum number of bits taken to encode

a transition probability between two states of $\mathcal{M}$, i.e. if $t(s,a,s') > 0$ then $t(s,a,s') \geq 2^{-h}$ and $t(s,a,s')$ can be represented as sum $\sum_{i=1}^{h} a_i \cdot 2^{-(i-1)}$ by bits $b_1 \cdots b_h$.

The following polynomial-time nondeterministic Turing machine N evaluates the process in the manner of **PP**. On input $\mathcal{M} = (S, s_0, A, O, t, o, r)$ and π, N guesses a trajectory $\theta = \sigma_1, \ldots, \sigma_{|\mathcal{M}|}$ of length $|\mathcal{M}|$ with $\sigma_1 = s_0$ and $\sigma_i \in S$, calculates its probability $\text{prob}_\pi(\mathcal{M}, \theta)$ and the sum of rewards $r_\pi(\mathcal{M}, \theta)$ collected on it. Because π is concise, this takes time polynomial in $|\mathcal{M}|$. Finally, on that guessed sequence N produces either accepting or rejecting computations. Let $G = \text{prob}_\pi(\mathcal{M}, \theta) \cdot 2^{h \cdot |\mathcal{M}|} \cdot |r_\pi(\mathcal{M}, \theta)|$. If $r_\pi(\mathcal{M}, \theta) > 0$, then G accepting computations are produced, and otherwise G rejecting computations are produced. It is clear that the number of accepting computations of N minus the number of rejecting computations of N equals the performance of $\mathcal{M}$ under π multiplied by $2^{h \cdot |\mathcal{M}|}$. Hence, N has more accepting than rejecting computations if and only if $\mathcal{M}$ has performance > 0 under π. □

This upper bound is strict, what is shown by proving that the evaluation problem is complete for the class **PP**.

Theorem 4.9 [2] *The concise policy evaluation problem for POMDPs is* **PP***-complete.*

Proof. We show that the problem is **PP**-hard. The problem MAJSAT – given a Boolean formula in 3CNF, is at least half of the assignments satisfying – is known to be **PP**-complete. We give a polynomial-time reduction from MAJSAT to our performance function.

Let ϕ be a formula in 3CNF with variables $x_1, \ldots, x_n$. We define a POMDP $\mathcal{M}(\phi)$ and a concise policy π as follows.

From its initial state, $\mathcal{M}(\phi)$ randomly chooses assignments subsequently to $x_1, \ldots, x_n$. These first n transitions are independent from the action chosen by the policy. Afterwards, $\mathcal{M}(\phi)$ runs deterministically successively through all clauses of ϕ and through all literals in the clauses. For each literal, the policy decides which value to assign to it. If finally the formula was satisfied by the policy, reward 1 is obtained, and if the formula is not satisfied, reward -1 is obtained. Note that $\mathcal{M}(\phi)$ has 2^n different trajectories.

The policy π must guarantee, that a trajectory gets a reward 1 if and only if the assignment, that was randomly chosen on the first steps of the trajectory, satisfies the formula. Therefore, we define π to choose action $\perp$ during the first $n+1$ steps (remind that during these steps the actions chosen by the policy do not influence the state transitions). After $n+1$ steps, the evaluation of ϕ begins. Now, we define π to choose the ith observation – which is the random assignment to variable x_i – when asked for an assignment to variable x_i. Since π is simply a selector function, it can be represented by a small circuit. $\mathcal{M}(\phi)$ has performance > 0 under π if and only if $\phi \in$ MAJSAT.

Formally, let ϕ consist of the m clauses $C_1, \ldots, C_m$, where clause $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ for literals $l_{i,j}$. Define $\mathcal{M}(\phi) = (S, s_0, A, O, t, o, r)$ with

$$\begin{aligned}
S &= \{s_0, s_{end}\} \cup \{[x_i = a] | i \in \{1,2,\ldots,n\}, a \in \{0,1\}\} \\
&\quad \cup \{l_{i,j} \mid i \in \{1,2,\ldots,n\}, j \in \{1,2,3\}\} \\
A &= \{\perp\} \cup \{x_i, \neg x_i \mid i \in \{1,2,\ldots,n\}\} \\
O &= S \\
o &= id \text{ (states and observations are equal)}
\end{aligned}$$

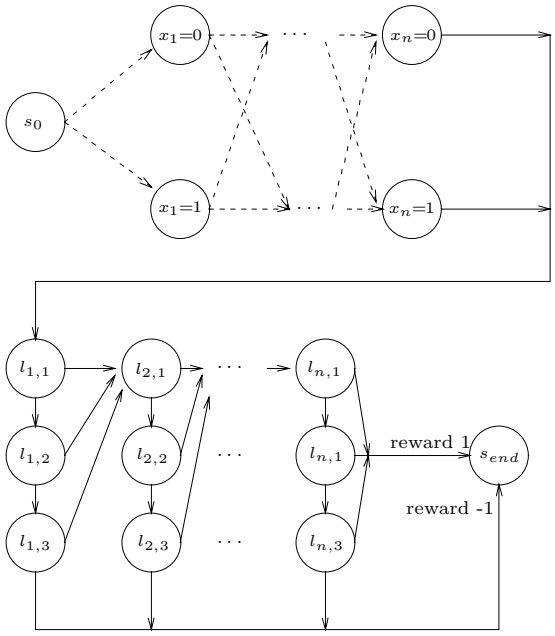


Figure 8: $\mathcal{M}(\phi)$

$$t(s,a,s') \quad = \quad \begin{cases} \frac{1}{2}, & \text{if } s = s_0, a = \bot, s' = [x_1 = b], b \in \{0,1\} \\ \frac{1}{2}, & \text{if } s = [x_i = b], a = \bot, s' = [x_{i+1} = b], \\ & \qquad i \in \{1,2,\ldots,n-1\}, \quad b,c \in \{0,1\} \\ 1, & \text{if } s = [x_n = b], a = \bot, s' = l_{1,1}, b \in \{0,1\} \\ 1, & \text{if } s = l_{i,j}, a = \overline{l_{i,j}}, s' = l_{i,j+1}) = 1, 1 \leq j < 3 \\ 1, & \text{if } s = l_{i,3}, a = \overline{l_{i,3}}, s' = \mathsf{s}_{end}, a = \overline{l_{i,3}} \\ 1, & \text{if } s = l_{i,j}, a = l_{i,j}, l_{i+1,1}, i < n \\ 1, & \text{if } s = l_{n,j}, a = l_{n,j}, s' = \mathsf{s}_{end} = 1 \\ 1, & \text{if } s = l_{n,j}, s' = \mathsf{s}_{end} = 1, j = 3 \\ 1, & \text{if } s = s' = \mathsf{s}_{end} \end{cases}$$

A non-zero reward is obtained only if the final state s_{end} is reached: if all clauses were satisfied, reward 1 is obtained; otherwise the reward is -1.

In Figure 8, $\mathcal{M}(\phi)$ is sketched. Dashed lines indicate transitions with probability $\frac{1}{2}$ under action $\bot$, solid lines indicate transitions with probability 1. A solid line leaving state $l_{i,j}$ to the right has action $l_{i,j}$, a solid line leaving it to the bottom has action $\overline{l_{i,j}}$. All rewards are 0, excepted transitions to the final state.

Let π be the following policy. If the last observation (i.e. the observation made in the state the process is actually in) is a literal $l_{i,j}$ which consists of variable x_k, then action x_k is applied if observation "$x_k = 1$" is contained in the history, and action $\neg x_k$ is applied if observation "$x_k = 0$" is contained in the history. Otherwise, action $\bot$ is applied. This policy can be represented by a circuit polynomial in the size of $|\mathcal{M}(\phi)|$.

$\mathcal{M}(\phi)$ under π has 2^n trajectories of equal probability. Each trajectory obtains either reward 1 or reward -1, and each trajectory obtains reward 1 if and only if the variable assignment chosen in the first steps satisfies ϕ. Therefore it follows that the performance of $\mathcal{M}(\phi)$ under π is greater than 0 if and only if $\phi \in$ MAJSAT. This proves the **PP**-hardness of the problem.

Together with Lemma 4.8 – which states that the problem is in **PP** – completeness follows. □

We now turn to history-dependent policies in general. For a UMDP, a history-dependent policy is the same as a time-dependent policy. Moreover, if the UMDP has only one action, the only short-term history-dependent policy for it can be calculated in space logarithmic in the size of the UMDP. Therefore, from Corollary 4.4 it follows that the history-dependent policy evaluation problem for UMDPs and for POMDPs is **PL**-hard. To show, that it is in **PL**, we can extend the method used in the proof of Theorem 4.5 for time-dependent policies. Given a POMDP $\mathcal{M} = (S, s_0, A, O, t, o, r)$ and a history-dependent policy π, we construct a new UMDP $\widehat{\mathcal{M}}$ with states $S \times O^{\leq |\mathcal{M}|}$, only one action a and transition probabilities $\hat{t}((s,\omega), a, (s', \omega o(s')) = t(s, \pi(\omega), s')$. Actually, the performance of $\mathcal{M}$ under π is the same as that of $\widehat{\mathcal{M}}$ under the stationary policy a. Because $(\widehat{\mathcal{M}}, a)$ can be computed in logarithmic space on input $(\mathcal{M}, \pi)$, it follows that the history-dependent policy evaluation problem for POMDPs logspace reduces to the stationary policy evaluation problem for POMDPs. The latter is in **PL** (Theorem 4.2), and because **PL** is closed downward under logspace reductions, the former is in **PL**, too.

Theorem 4.10 *The history-dependent policy evaluation problem for POMDPs is* **PL***-complete.*

Notice, that the complexity of the history-dependent policy evaluation problem is so low because history-dependent policies are so large.

4.2 Compressed Representations

Because the number of observations may be exponential in the representation size of a compressed POMDP, specifying a stationary policy is intractable. Moreover, an evaluation problem consisting of pairs of POMDPs and a very large policy description has very low complexity, since the complexity is measured in the size of those pairs. The only exception are policies for compressed UMDPs. Each stationary policy consists of one action only (i.e. a mapping from the only observation class to an action), and each time-dependent policy consists of a sequence of actions. Those policies can be specified using only polynomially many bits in the description size of the compressed UMDP and hence do not "pathologically" decrease the complexity of evaluation problems. The complexity for these problems is intermediate between those for flat POMDPs and for succinct POMDPs.

Remember that **#P** is the class of functions f for which there exists a nondeterministic polynomial-time bounded Turing machine N such that $f(x)$ equals the number of accepting computations of N on x. Every polynomial-time computable integer function is in **FP**, and every function in **FP** can be calculated in polynomial space. Hence, **FP** lies between polynomial time and polynomial space like **PP** lies between **P** and **PSPACE**. The matrix powering problem for integer matrices with "small" entries is in **#P**, using the same proof idea as for Lemma 4.1.

Lemma 4.11 [3] *Let T be a $2^m \times 2^m$ matrix of nonnegative integers, each consisting of m bits. Let T be represented by a Boolean circuit C with $2m + \lceil \log m \rceil$ input gates, such that $C(a, b, r)$ outputs the r-th bit of $T_{(a,b)}$. For $1 \leq i, j \leq 2^m$, and $0 \leq s \leq m$, the function mapping (C, s, i, j) to $(T^s)_{(i,j)}$ is in* **#P***.*

As a consequence, the complexity of the policy evaluation problems can be shown to be intermediate between **NP** and **PSPACE**.

Theorem 4.12 [3] *The stationary and time-dependent policy evaluation problems for compressed UMDPs are* **PP***-complete.*

The proof, which relies on Lemma 4.11, is similar to that of Theorem 4.5.

Note that **PP** is apparently much more powerful than **NP**, although it is contained in **PSPACE**. Once again, we see that limiting the type of rewards can considerably simplify the computation.

Theorem 4.13 [3] *The stationary and time-dependent policy evaluation problems for compressed UMDPs with nonnegative rewards are* **NP***-complete.*

Proof. Remember that the horizon is roughly the size of the input, so that a trajectory can be guessed in polynomial time and then checked for consistency, positive probability, and positive reward.

The proof of **NP**-hardness is a reduction from SAT. Given a Boolean formula ϕ with m variables, we create an MDP with $2^m + 2$ states $\{s_0, s_1\} \cup \{s_w \mid w \in \{0,1\}^m\}$ and one action, a. The state s_0 is the initial state, state s_1 is a final sink state, and each of the other 2^m states represents an assignment to the m variables. From s_0, all of these 2^m states are accessed on action a, each with equal probability 2^{-m} and without reward. In each of the "assignment states," reward 1 is obtained on action a, if the assignment satisfies ϕ, and reward 0 is obtained otherwise. From each of the "assignment states," the sink state is accessed with probability 1. The process remains in the sink state without rewards. The unique policy $\pi = a$ has performance > 0 if and only if there is a satisfying assignment for ϕ. □

Circuits for concise policies take as input a history in form of a sequence of binary encodings of observations. Therefore, even for compressed POMDPs short-term concise policies are tractably representable. Surprisingly, the complexity of the concise policy evaluation problem does not depend on whether the POMDP is flat or compressed.

Theorem 4.14 [2] *The concise policy evaluation problem for compressed POMDPs is* **PP***-complete.*

Proof. The proof is essentially the same as for Theorem 4.9. To prove containment in **PP**, the nondeterministic algorithm given in the proof of Theorem 4.8 works for compressed POMDPs, too. Hardness for **PP** follows straightforwardly from hardness for flat POMDPs (Theorem 4.9). □

5 Short-Term Policy Existence for Flat POMDPs

The short-term policy existence problem asks whether a given POMDP $\mathcal{M}$ has positive performance under any policy with horizon $|\mathcal{M}|$. Unlike policy evaluation, it is feasible to consider whether a good enough history-dependent policy exists. We will see that in order to answer the existence problem for history-dependent policies, it is not necessary to specify a policy, and therefore the problem does not become automatically intractable. Instead, structural properties of POMDPs are used. Also unlike the policy evaluation problem, the observability of the POMDP has an effect on the complexity. Hence, we have (at least) three parameters – type of policy, type of observability, type of representation – which determine the complexity. The complexity trade-offs between these parameters are interesting, and it is important for a system designer to know them.

5.1 Unobservable POMDPs

For UMDPs, only stationary and time-dependent policies are of interest. Given a flat UMDP $\mathcal{M}$, an arbitrary stationary or time-dependent short-term policy has size at most $|\mathcal{M}|^3$, and hence it can be guessed non-deterministically in polynomial time. Once we have $\mathcal{M}$ and a policy, we can check

in **PL** whether the policy's performance is greater 0 (Theorem 4.5). Because **PL** is a subclass of **P**, the existence problem can be decided by a polynomial-length "guess" and a polynomial-time "check". This means, that **NP** is an upper bound for the complexity of the policy existence problem for UMDPs. On the other hand, **PL** looks like a natural lower bound for the problem, because finding a policy should not be easier than evaluating one. The following results show that this lower resp. this upper bound are the exact complexity of the stationary resp. time-dependent policy existence problem for UMDPs.

Theorem 5.1 [3] *The stationary policy existence problem for UMDPs is* **PL***-complete.*

Proof. First we show that the problem is in **PL**. Let $\mathcal{M}$ be a UMDP with set of actions A. Because $\mathcal{M}$ is unobservable, every stationary policy for $\mathcal{M}$ is a constant function $a \in A$. Then there exists a policy under which $\mathcal{M}$ has performance greater than 0 if and only if for some $a \in A$, $\mathcal{M}$ under a has performance greater than 0. For any $a \in A$, let $\mathcal{M}_a$ be the same as $\mathcal{M}$ but only action a has transition probabilities > 0. The set $M_A = \{(\mathcal{M}_a, a) \mid a \in A\}$ can be computed in logarithmic space on input $\mathcal{M}$. M_A consists of instances of the stationary policy evaluation problem. It contains an instance that is *in* the stationary policy evaluation problem if and only if for some $a \in A$, $\mathcal{M}$ under a has performance greater than 0. Thus the stationary policy existence problem for UMDPs reduces disjunctively in logarithmic space to the stationary policy *evaluation* problem for UMDPs. From Lemma 4.2 and the closure of **PL** under logarithmic space disjunctive reductions (see Allender and Ogihara (1996)), it follows that the policy existence problem is in **PL**.

In order to show **PL**-hardness, note that for POMDPs with only one action, there is no difference between the complexity of the policy evaluation problem and that of the policy existence problem. In the proof of Lemma 4.3, every **PL** computation was logspace reduced to a UMDP with exactly one action only. This proves **PL**-hardness of the policy existence problem, too. □

A time-dependent policy for a UMDP allows one to choose one action for each step, instead of one action which is performed on all steps of a stationary policy. This alters the complexity of the policy existence problem from **PL** to **NP** (assuming those classes are distinct). (See also Papadimitriou and Tsitsiklis (1987).)

Theorem 5.2 [3] *The time-dependent policy existence problem for UMDPs is* **NP***-complete.*

Proof. That the problem is in **NP** follows from the fact that a policy with performance > 0 can be guessed and checked in polynomial time. **NP**-hardness follows from the following reduction from 3SAT, the set of satisfiable Boolean formulas in conjunctive normal form with clauses of at most three literals. Given such a formula ϕ with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$, we construct an UMDP that evaluates the clauses in parallel. At the first step, the UMDP chooses randomly and independent on the action one of the m clauses. At step $i+1$, the policy determines an assignment to variable x_i. The process checks, whether this assignment satisfies the clause it is in. Because the process is unobservable, the policy must assign the same value to all appearances of this variable. If a clause was satisfied, it will gain reward 1, if not, the reward will be $-m$, where m is the number of clauses of the formula. Therefore, if all m clauses are satisfied, the time-dependent value of the UMDP is positive, otherwise negative.

We formally define the reduction. We say that *variable* x_i 1*-appears in clause* C_j, if $x_i \in C_j$, and x_i 0*-appears*, if $\neg x_i \in C_j$. In a clause C_j a variable x_i also may not appear at all. Define the UMDP $\mathcal{M}(\phi) = (S, s_0, A, t, r)$ where

$$\begin{aligned} S &= \{(i,j) \mid 1 \le i \le n, 1 \le j \le m\} \cup \{s_0, \mathsf{t}, \mathsf{f}\} \\ A &= \{0, 1\} \end{aligned}$$

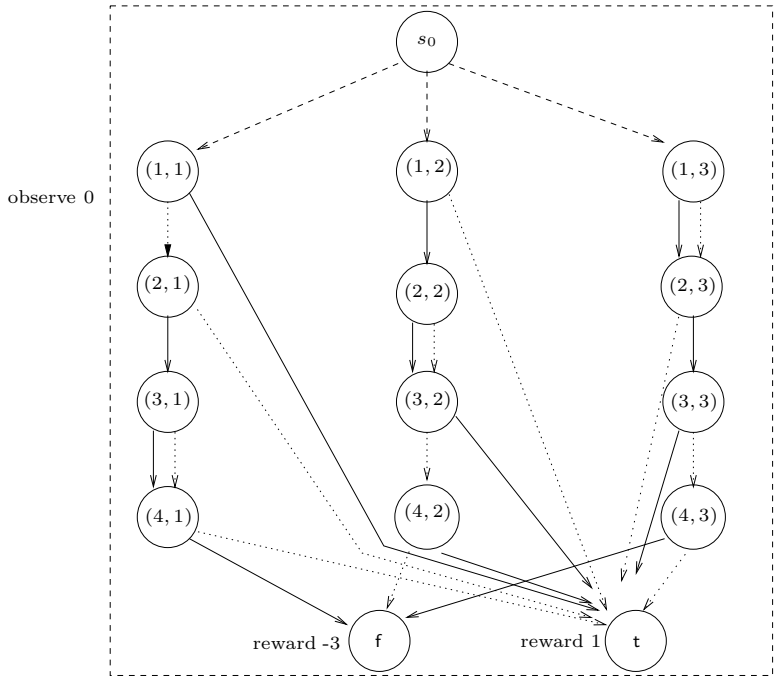


Figure 9: $\mathcal{M}(\phi)$ for $\phi = (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$.

$$
t(s,a,s') = \begin{cases}
\frac{1}{m}, \text{ if } s = s_0, a = 0, s' = (1,j), 1 \le j \le m \\
1\,, \text{ if } s = (i,j), s' = \mathsf{t},\ x_i \text{ } a\text{-appears in } C_j \\
1\,, \text{ if } s = (i,j), s' = (i+1,j), i < n, \\
\qquad\qquad x_i \text{ does not } a\text{-appear in } C_j \\
1\,, \text{ if } s = (n,j), s' = \mathsf{f},\ x_n \text{ does not } a\text{-appear in } C_j \\
1\,, \text{ if } s = s' = \mathsf{f} \text{ or } s = s' = \mathsf{t}, a = 0 \text{ or } a = 1 \\
0\,, \text{otherwise}
\end{cases}
$$

$$
r(s,a) = \begin{cases}
1\ \ , \text{if } t(s,a,\mathsf{t}) > 0 \text{ and } s \neq \mathsf{t} \\
-m, \text{if } t(s,a,\mathsf{f}) > 0 \text{ and } s \neq \mathsf{f} \\
0\ \ , \text{otherwise.}
\end{cases}
$$

The correctness of the reduction follows by the above discussion.

Figure 9 sketches the construction of $\mathcal{M}(\phi)$ for $\phi = (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$. The dashed lines from the initial state indicate a transition with probability $\frac{1}{3}$ under action 0. Solid lines indicate transitions on action 1, and dotted lines transitions on action 0. All actions have reward 0 unless a reward is indicated. □

5.2 Fully-Observable POMDPs

The computation of optimal policies for fully-observable POMDPs is a well-studied optimization problem. The maximal performance of any infinite-horizon stationary policy for a fully-observable Markov decision process can be solved by linear programming techniques in polynomial time. Dynamic programming can be used for the finite-horizon time-dependent and history-dependent policy cases.

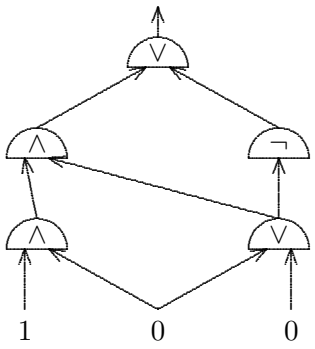


Figure 10: A Boolean circuit with AND, OR and NOT gates, and input 100.

The exact complexity of the stationary policy existence problem for fully-observable POMDPs with finite horizon is not known. From Papadimitriou and Tsitsiklis (1987) it follows that it is **P**-hard, and it is easily seen to be in **NP**.

Theorem 5.3 [3] *The stationary policy existence problem for MDPs is* **P***-hard and in* **NP***.*

The proof is a straightforward modification of the proof of Theorem 1 by Papadimitriou and Tsitsiklis (1987). The **P**-hardness is shown by a reduction from the **P**-complete *circuit value problem* CVP: given a Boolean circuit C and an input x, does C on input x output value 1? Circuit C has AND, OR, and NOT gates, and inputs according to x at the input gates. An example of such a circuit is given in Figure 10. Papadimitriou and Tsitsiklis use the *monotone circuit value problem*, which considers only circuits without NOT gates. For later use of the proof technique we need to deal with circuits that also have NOT gates. Therefore, we present the proof here.
Proof. That the problem is in **NP** can be seen using the "guess-and-check" approach: guess a stationary policy and check whether its performance is greater than 0.

Now we prove that the problem is **P**-hard. We give a reduction from the **P**-complete problem CVP. A Boolean circuit and its input can be seen as a directed acyclic graph. Each node represents a gate, and every gate has one of the types AND, OR, NOT, 0 or 1. The gates of type 0 or 1 are the input gates, which represent the bits of the fixed input x to the circuit. Input gates have indegree 0. All NOT gates have indegree 1, and all AND and OR gates have indegree 2. There is one gate having outdegree 0. This gate is called the output gate, from which the result of the computation of circuit C on input x can be read.

From such a circuit C, a MDP $\mathcal{M}$ can be constructed as follows. Because the basic idea of the construction is very similar to one shown in Papadimitriou and Tsitsiklis (1987), we leave out technical details. For simplicity, assume that the circuit has no NOT gates. Each gate of the circuit becomes a state of the MDP. The start state is the output gate. Reverse all edges of the circuit. Hence, a transition in $\mathcal{M}$ leads from a gate in C to one of its predecessors. A transition from an OR gate depends on the action and is deterministic. On action 0 its left predecessor is reached, and on action 1 its right predecessor is reached. A transition from an AND gate is probabilistic and does not depend on the action. With probability $\frac{1}{2}$ the left predecessor is reached, and with probability $\frac{1}{2}$ the right predecessor is reached. If an input gate with value 1 is reached, a positive reward is gained, and if an input gate with value 0 is reached, a high negative reward is gained,

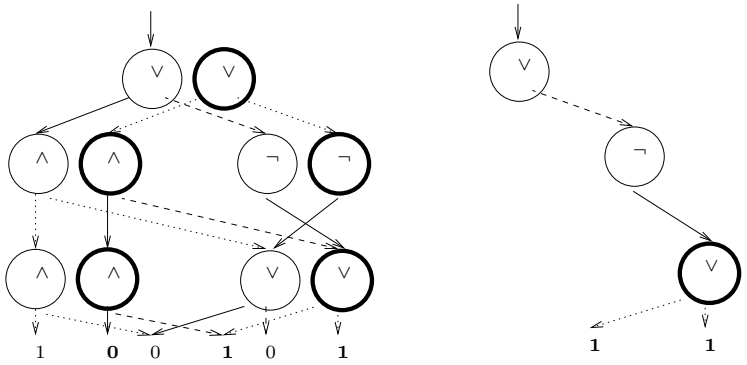


Figure 11: The MDP to which the circuit from Figure 10 is reduced to, and the trajectories according to an optimal policy for the MDP.

---

which makes the total expected reward negative. If $C(x) = 1$, then the actions can be chosen in the OR gates that every trajectory reaches an input gate with value 1, and vice versa. Hence, the MDP has a positive value if and only if $C(x) = 1$.

If the circuit also has NOT gates, one needs to remember the parity of the number of NOT gates on every trajectory. If the parity is even, everything goes as described above. If the parity is odd, then the role of AND and OR gates is switched, and the role of 0 and 1 gates is switched. If a NOT gate is reached, the parity bit is flipped. For every gate in the circuit, we now take two MDP states: one for even and one for odd parity. The transition probability is extended accordingly. Now we have to adjust the reward function. If an input gate with value 1 is encountered on a trajectory where the parity of NOT gates is even, then reward $-2^{|C|}$ is obtained, where $|C|$ is the size of circuit C. The same reward is obtained if an input gate with value 0 is encountered on a trajectory where the parity of NOT gates is odd. All other trajectories obtain reward 1.

There are at most $2^{|C|}$ trajectories for each policy. If $C(x) = 1$, then a policy exists that chooses the right actions for all the gates. This policy has performance 1. If a policy with performance 1 exists, then one can similarly conclude that $C(x) = 1$. Every policy with performance other than 1, has performance at most $\frac{2^{|C|}-1}{2^{|C|}} - 1$, what is negative performance. Therefore, $C(x) = 1$ if and only if a stationary policy with performance > 0 exists. □

In Figure 11 an example MDP to which the circuit from Figure 10 is transformed to is given. Every gate of the circuit is transformed to two states of the MDP, one copy for *even* parity of NOT gates passed on that trajectory (indicated by a thin outline of the state) and one copy for *odd* parity of NOT gates passed on that trajectory (indicated by a thick outline of the state respectively by a boldface value of states corresponding to input gates). A solid arrow indicates the outcome of action "choose the left predecessor", and a dashed arrow indicates the outcome of action "choose the right predecessor". Dotted arrows indicate a transition with probability $\frac{1}{2}$ on any action. The circuit in Figure 10 has value 1. The policy, which chooses the right predecessor in the starting state, yields trajectories which all end in an input gate with value 1 and which therefore obtains the optimal value.

Notice that the type of policy does not matter in the proof of Theorem 5.3. This yields that **P**-hardness also holds for the time-dependent and for the history-dependent policy existence problem for MDPs. On the other hand, it is well known that the (finite-horizon) value of an MDP is achieved by a time-dependent policy, which can be found in polynomial time. Hence, these policy existence

problems are **P**-complete (see also Theorem 1 in Papadimitriou and Tsitsiklis (1987)).

Theorem 5.4 *The time-dependent and history-dependent policy existence problems for MDPs are* **P***-complete.*

Hence, considerations of short-term policy existence problems for MDPs leave an open problem: what is the exact complexity of the stationary policy existence problem?

5.3 POMDPs

For the problems with UMDPs and MDPs considered up to now, only the time-dependent policy existence problem for UMDPs is complete for the class that is a straightforward upper bound on the problem complexity – namely **NP**. The other problems have (or seem to have) lower complexity. We will see now that for POMDPs, all problems are complete for their straightforward upper bounds.

Theorem 5.5 [3] *The stationary policy existence problem for POMDPs is* **NP***-complete.*

Proof. Membership in **NP** is straightforward, because a policy can be guessed and evaluated in polynomial time. To show **NP**-hardness, we reduce 3SAT to it. Let ϕ be a formula with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$, where clause $C_j = (l_{v(1,j)} \vee l_{v(2,j)} \vee l_{v(3,j)})$ for $l_i \in \{x_i, \neg x_i\}$. We say that variable x_i 0*-appears (resp.* 1*-appears) in* C_j if $\neg x_i$ (resp. x_i) is a literal in C_j. Without loss of generality, we assume that every variable appears at most once in each clause. The idea is to construct a POMDP $\mathcal{M}(\phi)$ having one state for each appearance of a variable in a clause. The set of observations is the set of variables. Each action corresponds to an assignment of a value to a variable. The transition function is deterministic. The process starts with the first variable in the first clause. If the action chosen in a certain state satisfies the corresponding literal, the process proceeds to the first variable of the next clause, or with reward 1 to the final state, if all clauses were considered. If the action does not satisfy the literal, the process proceeds to the next variable of the clause, or with reward 0 to the final state. The partition of the state space into observation classes guarantees that the same assignment is made for every appearance of the same variable. Therefore, the maximal performance of $\mathcal{M}(\phi)$ equals 1 if ϕ is satisfiable, and it equals 0 otherwise.

Formally, from ϕ, we construct a POMDP $\mathcal{M}(\phi) = (S, s_0, A, O, t, o, r)$ with

$$
\begin{array}{rcl}
S & = & \{(i,j) \mid 1 \le i \le n, 1 \le j \le m\} \cup \{\mathsf{f}, \mathsf{t}\} \\
s_0 & = & (v(1,1),1) \\
A & = & \{0,1\} \\
O & = & \{x_1, \ldots, x_n, \mathsf{f}, \mathsf{t}\}
\end{array}
$$

$$
o(s) = \begin{cases} x_i, & \text{if } s = (i,j) \\ \mathsf{t}, & \text{if } s = \mathsf{t} \\ \mathsf{f}, & \text{if } s = \mathsf{f} \end{cases}
$$

$$
r(s,a) = \begin{cases} 1, & \text{if } t(s,a,\mathsf{t}) = 1, s \neq \mathsf{t} \\ 0, & \text{otherwise} \end{cases}
$$

$$t(s,a,s') \quad = \quad \begin{cases} 1, & \text{if } s = (v(i,j),j), s' = (1,j+1), j < m, 1 \le i \le 3, \\ & \qquad \text{and } x_{v(i,j)} \text{ } a\text{-appears in } C_j \\ 1, & \text{if } s = (v(i,m),m), s' = \mathsf{t}, 1 \le i \le 3, \\ & \qquad \text{and } x_{v(i,m)} \text{ } a\text{-appears in } C_m \\ 1, & \text{if } s = (v(i,j),j), s' = (v(i+1,j),j), 1 \le i < 3, \\ & \qquad \text{and } x_{v(i,j)} \text{ } (1-a)\text{-appears in } C_j \\ 1, & \text{if } s = (v(3,j),j), s' = \mathsf{f}, \\ & \qquad \text{and } x_{v(3,j)} \text{ } (1-a)\text{-appears in } C_j \\ 1, & \text{if } s = s' = \mathsf{f} \text{ or } s = s' = \mathsf{t} \\ 0, & \text{otherwise} \end{cases}$$

Note that all transitions in $\mathcal{M}(\phi)$ either have probability 1 or 0 and hence are deterministic, and every trajectory has reward 0 or 1. Each assignment to ϕ can be interpreted as a stationary policy for $\mathcal{M}(\phi)$ and vice versa: the value assigned to a variable equals the action assigned to an observation. Policies under which $\mathcal{M}(\phi)$ has performance 1 correspond to satisfying assignments for ϕ, and vice versa. Therefore, ϕ is satisfiable if and only if there exists a stationary policy under which $\mathcal{M}(\phi)$ has performance > 0.

Figure 12 sketches the construction of $\mathcal{M}(\phi)$ for $\phi = (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$. Note that ϕ is the same as in Figure 9.

Solid lines indicate transitions on action 1, and dotted lines are transitions on action 0. All actions have reward 0 unless reward 1 is indicated. □

Because in the proof of Theorem 5.5, the POMDP constructed to show **NP**-hardness is deterministic, and every trajectory gets reward 0 or 1 only in one step, we get the same completeness for POMDPs with nonnegative rewards (see also Littman (1994) for a comparable result).

Corollary 5.6 *The stationary policy existence problem for POMDPs with nonnegative rewards is* **NP***-complete.*

Since UMDPs are POMDPs, the same hardness proof as in Theorem 5.2 applies for the time-dependent policy existence problem for POMDPs.

Corollary 5.7 *The time-dependent policy existence problem for POMDPs is* **NP***-complete.*

In the proof of Theorem 5.5, negative rewards are essential. With nonnegative rewards, the problem is easily seen to be in **NL**: just find a path through the UMDP that leads to a state which allows positive reward on some action. In the same way, the **NL**-hard graph reachability problem can be shown to reduce to the policy existence problem. The same technique can be applied for history-dependent policies.

Theorem 5.8 [3] *The time-dependent and history-dependent policy existence problems for UMDPs and POMDPs both with nonnegative rewards are* **NL***-complete.*

For (fully-observable) MDPs the optimal time-dependent and the optimal history-dependent policy have the same performance. Hence, the respective policy existence problems have the same complexity. For POMDPs the complexity is different. Instead, POMDPs seem to obtain their expressive power by history-dependent policies.

Theorem 5.9 [3] *The history-dependent policy existence problem for POMDPs is* **PSPACE***-complete.*

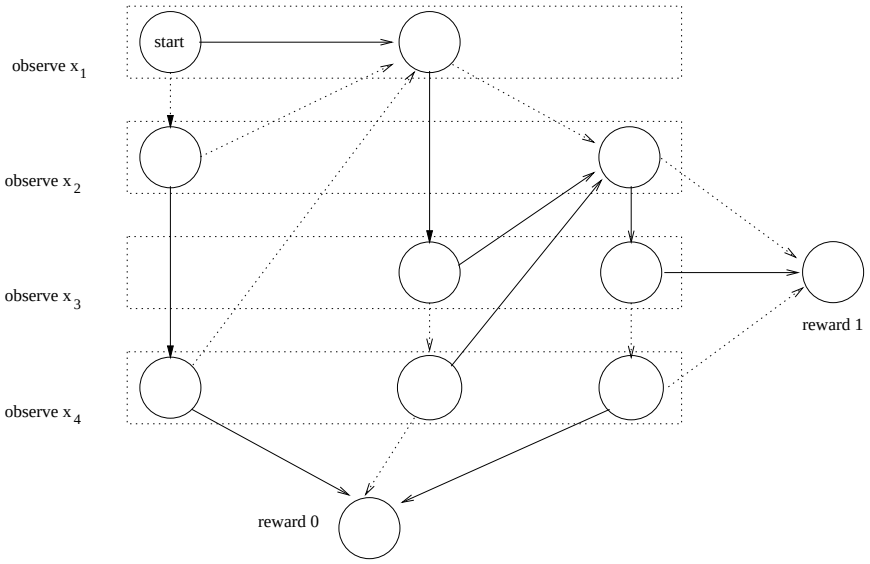


Figure 12: $\mathcal{M}(\phi)$ for $\phi = (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$.

In [3] (Mundhenk, Goldsmith, Lusena, and Allender 2000), this Theorem is proven using a technique adopted from Papadimitriou and Tsitsiklis (1987). In fact, our approach that the problem is in **PSPACE** uses the idea from Papadimitriou and Tsitsiklis (1987). To prove **PSPACE**-hardness, we use a new and more powerful technique developed from a proof in [2] (Mundhenk 2000) that also finds applications non-approximability (see Theorem 5.16).

Proof. We show that this problem is in **NPSPACE** what equals **PSPACE**. An entire history-dependent policy for a horizon n equal to the size of the input may have size exponential in n, and therefore it cannot be completely specified within polynomial space. However, a policy can be guessed stepwise.

The set of possible histories to consider forms a tree of depth n. A policy can be specified by labeling the edges of that tree with actions. Such a policy can be guessed one branch at a time; in order to maintain consistency, one need only keep track of the current branch.

To evaluate a policy, for each branch through the policy tree, one must compute the value of each node under that policy. The probability of each transition can be represented with n bits; the product of n such requires n^2 bits. Rewards require at most n bits, and are accumulated (potentially) at each transition. The total reward for a given trajectory, therefore, requires at most n^3 bits.

There are at most n states, so there are at most $n^n = 2^{n \cdot \log n}$ trajectories. The value of each is bounded by 2^{n^3}, so the sum is bounded by $2^{n^3 + n \cdot \log n}$, and thus can be represented by polynomially many bits.

To show **PSPACE**-hardness, we give a reduction from the **PSPACE**-complete set QBF. For an instance ϕ of QBF, where ϕ is a 3CNF formula with n variables $x_1, \ldots, x_n$, we construct a POMDP $\mathcal{M}(\phi)$ as follows. $\mathcal{M}(\phi)$ has three stages. The first stage consists of one random step. The process randomly chooses one of the variables and an assignment to it, and stores the variable and the assignment. This means, from the initial state s_0, one of the states "$x_i = b$" $(1 \leq i \leq n, b \in \{0, 1\})$ is reached, each with probability $1/(2n)$. The partial observability of the process is made use of in the way, that it becomes unobservable which variable assignment was stored by the process.

Whenever the variable appears the assignment to which is stored by the process, the process checks that the initially fixed assignment is chosen again. If the policy gives a different assignment during the first phase, the process halts with reward 0. If this happens during the second phase, a very high negative reward is obtained which ensures that the process has a negative performance. If eventually the whole formula is passed, reward 1 or reward $-2n \cdot 2^m$ (for m equal the number of universally quantified variables of ϕ) is obtained dependent on whether the formula was satisfied or not.

The second stage starts in each of the states "$x_i = b$" and has n steps, during which an assignment to all variables $x_1, x_2, \ldots, x_n$, one after the other, is fixed. If a variable x_i is existentially quantified, then the assignment depends on the action chosen by the policy. If a variable x_i is universally quantified, then the assignment is randomly chosen by the process, independent on the action of the policy. In the second stage, it is observable, which assignment was made to every variable. If the variable assignment from the first stage does not coincide with the assignment made to that variable during the second stage, the trajectory on which that happens ends in an error state that yields reward 0. Starting from state "$x_i = b$", there are at most 2^m different trajectories up to the end of the second stage. On each of these trajectories, the last n observations are the assignments to the variables $x_1, x_2, \ldots, x_n$. These assignments are observable by the policy. Notice that x_i is assigned b.

In the third stage, it is checked whether ϕ is satisfied by that trajectories assignment. The process passes sequentially through the whole formula and asks one literal after the other in each clause after the other for an assignment to the respective variable. It must be excluded that the policy cheats, i.e. it answers with different assignments at different appearances of the same variable. Whenever the variable appears the assignment to which is stored by the process during the first stage, the process checks that the stored assignment is chosen again. A very high negative reward is obtained otherwise, which ensures that the process has a negative performance. If eventually the whole formula is passed, reward 1 or reward -1 is obtained dependent on whether the formula was satisfied or not.

Now we go into construction details. The set of actions of $\mathcal{M}(\phi)$ is $A = \{0, 1\}$, where 0 and 1 will be used as assignments to variables. The set of observations is $O = \{*, 0, 1, x_1, \ldots, x_n\}$, where 0 and 1 mean that the respective value is assigned to a variable, and the variables mean that an assignment to it is asked for. The initial state of $\mathcal{M}(\phi)$ is s_0, from which on every action one of the states in $\{[x_c = b] \mid 1 \leq c \leq n, b \in \{0, 1\}\}$ is reached with equal probability, i.e. $t(s_0, a, [x_c = b]) = \frac{1}{2n}$ for every action a and all appropriate c, b. All states $[x_c = b]$ have the same observation $*$, and all rewards for the first transition are 0. This describes the first stage.

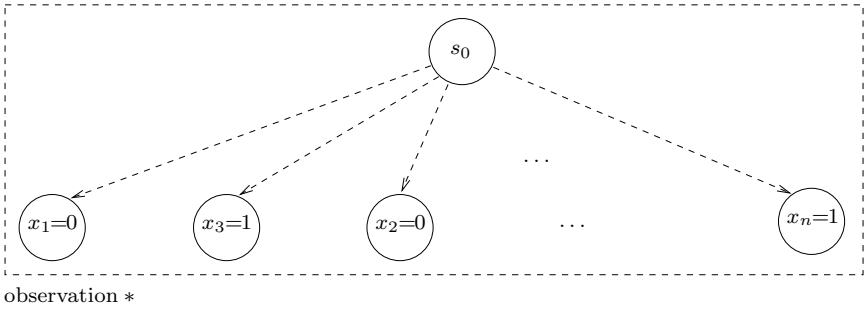


Figure 13: The first stage of $\mathcal{M}(\phi)$.

Each state $[x_c = b]$ can be seen as initial state of a subprocess $\mathcal{M}_{c,b}$ (for $1 \leq c \leq n$ and $b \in \{0,1\}$). Each of these subprocesses checks whether all assignments made to variable x_c during the process are equal b. If it is detected, that this is not the case, a high "fee" has to be paid (i.e. a negative reward). We make it unobservable in which of the subprocesses a state is.

Each of the subprocesses $\mathcal{M}_{c,b}$ consists of the second and the third stage described above. It starts in state "$x_c = b$". During the second stage, the variable assignment is fixed. This part of $\mathcal{M}_{c,b}$ is called $\mathcal{A}_{c,b}$. The assignments are observable. If b is not assigned to x_c, then the process halts without any reward. In the third stage, called $\mathcal{C}_{c,b}$, the formula is passed sequentially and it is checked, whether the policy assigns b to x_c, and whether all clauses are satisfied. In each state of $\mathcal{C}_{c,b}$ it is observable at which point of the formula the process is.

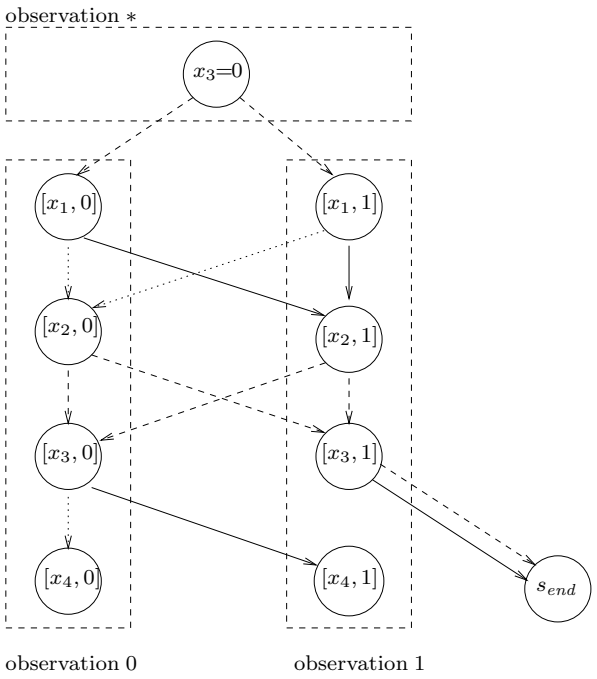


Figure 14: The second stage of $\mathcal{M}(\phi)$: $\mathcal{A}_{3,0}$ for the quantifier prefix $\forall x_1 \exists x_2 \forall x_3 \exists x_4$.

We now describe $\mathcal{A}_{c,b}$, the subprocesses fixing an assignment to the variables which assigns $a \in \{0,1\}$ to x_j $(1 \leq j \leq n)$. Its state set is

$$S^{\mathcal{A}}_{c,b} = \{[x_c = b], \mathsf{s}_{end}\} \cup \{[x_j = a]_{c,b} \mid j \in \{1, 2, \ldots, n\}, a \in \{0,1\}\}$$

where $[x_c = b]$ can be seen as its initial state, $[x_j = a]_{c,b}$ means that x_j is assigned a, and s_{end} is the final sink state of the process $\mathcal{M}(\phi)$. The observations reflect the assignments made, namely

- $o([x_c = b]) = o(\mathsf{s}_{end}) = *$ (no "specific" observation in initial or sink state)
- $o([x_j, a]_{c,b}) = a$ for $1 \leq j \leq n$; $a \in \{0,1\}$ (observe the assignment a made to x_j)

The state transitions are directed by actions or by random and determine an assignment to the variables.

$$t([x_c = b], a, [x_1, a']_{c,b}) \quad =$$

$$\begin{cases} 1, & \text{if } x_1 \text{ is existentially quantified, and } a = a' \\ 0, & \text{if } x_1 \text{ is existentially quantified, and } a \neq a' \\ \frac{1}{2}, & \text{if } x_1 \text{ is universally quantified} \end{cases}$$

$$t([j,d]_{c,b}, a, [x_{j+1}, a']_{c,b}) \quad = \quad \begin{cases} 0, & \text{if } j = c \text{ and } d \neq b \\ 1, & \text{if } x_{j+1} \text{ is existentially quantified, and } a = a' \\ 0, & \text{if } x_{j+1} \text{ is existentially quantified, and } a \neq a' \\ \frac{1}{2}, & \text{if } x_{j+1} \text{ is universally quantified} \end{cases}$$

$$t([x_c = 1-b]_{c,b}, a, \mathsf{s}_{end}) \quad = \quad 1$$

All transitions have reward 0.

The structure of $\mathcal{A}_{k,a}$ is for $\phi = \forall x_1 \exists x_2 \forall x_3 \exists x_4 (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$ is sketched in Figure 14. There, solid lines indicate deterministic transitions under action 1, dotted lines indicate deterministic transitions under action 0, and dashed lines indicate transitions with probability $\frac{1}{2}$ on any action.

$\mathcal{A}_{c,b}$ is linked to $\mathcal{C}_{c,b}$ in that from states $[x_n, a]_{c,b}$ the initial state $[l_{1,1}]_{c,b}$ of $\mathcal{C}_{c,b}$ is reached with probability 1 on action $\perp$, unless $i = n$ and $a \neq c$.

The subprocess $\mathcal{C}_{c,b}$ sequentially "asks" for assignments to the variables of ϕ as they appear in the literals. It is essentially the same process as defined in the proof of Theorem 5.5, but whenever an assignment to a literal containing x_c is asked for, a huge negative reward is obtained in case that x_c does not get assignment b.

Let ϕ consist of clauses $C_1, \ldots, C_m$, where clause $C_j = (l_{v(1,j)} \vee l_{v(2,j)} \vee l_{v(3,j)})$ for $l_i \in \{x_i, \neg x_i\}$. We say that variable x_i 0-*appears (resp.* 1-*appears) in* C_j if $\neg x_i$ (resp. x_i) is a literal in C_j.

$$S^{\mathcal{C}}_{c,b} \quad = \quad \{(i,j)_{c,b} \mid 1 \leq i \leq n, 1 \leq j \leq m\} \cup \{\mathsf{f}, \mathsf{t}\}$$

State $(v(1,1),1)_{c,b}$ (corresponding to the first literal in the first clause of ϕ) can be seen as initial state of $\mathcal{C}_{c,b}$. It is reached from the last states $[x_n, 0]_{c,b}$ and $[x_n, 1]_{c,b}$ of $\mathcal{A}_{c,b}$ on any action, unless $c = n$.

$$t([x_n, d]_{c,b}, a, (v(1,1),1)_{c,b}) \quad = \quad \begin{cases} 0, & \text{if } n = c \text{ and } a \neq b \\ 1, & \text{otherwise} \end{cases}$$

If $s = (c,j)_{c,b}$ and $a \neq b$, then $t(s, a, \mathsf{s}_{end}) \quad = \quad 1$

If $s \neq (c,j)_{c,b}$, or $a = b$, then

$$
t(s,a,s') \quad = \quad \begin{cases} 1, & \text{if } s = (v(i,j),j)_{c,b}, s' = (1,j+1)_{c,b}, j < m, \\ & \qquad 1 \le i \le 3, \text{ and } x_{v(i,j)} \ a\text{-appears in } C_j \\ 1, & \text{if } s = (v(i,j),j)_{c,b}, s' = (1,j+1)_{c,b}, j < m, \\ & \qquad 1 \le i \le 3, \text{ and } x_{v(i,j)} \ a\text{-appears in } C_j \\ 1, & \text{if } s = (v(i,m),m)_{c,b}, s' = \mathsf{t}, 1 \le i \le 3, \\ & \qquad \text{and } x_{v(i,m)} \ a\text{-appears in } C_m \\ 1, & \text{if } s = (v(i,j),j)_{c,b}, s' = (v(i+1,j),j)_{c,b}, 1 \le i < 3, \\ & \qquad \text{and } x_{v(i,j)} \ (1-a)\text{-appears in } C_j \\ 1, & \text{if } s = (v(3,j),j)_{c,b}, s' = \mathsf{f}, \\ & \qquad \text{and } x_{v(3,j)} \ (1-a)\text{-appears in } C_j \\ 1, & \text{if } s = s' = \mathsf{f} \text{ or } s = s' = \mathsf{t} \\ 0, & \text{otherwise} \end{cases}
$$

$$
r(s,a) \quad = \quad \begin{cases} 1, & \text{if } t(s,a,\mathsf{t}) = 1, s \neq \mathsf{t} \\ -2n \cdot 2^m, & \text{if } t(s,a,\mathsf{f}) = 1, s \neq \mathsf{t} \\ -2n \cdot 2^m, & \text{if } t(s,a,\mathsf{s}_{end}) = 1, s \neq \mathsf{s}_{end} \\ 0, & \text{otherwise} \end{cases}
$$

$$
o(s) \quad = \quad \begin{cases} x_i, & \text{if } s = (i,j)_{c,b} \\ *, & \text{if } s \in \{\mathsf{s}_{end}, \mathsf{f}, \mathsf{t}\} \end{cases}
$$

Finally, $\mathcal{M}(\phi)$ consists of the first stage and the union of all processes $\mathcal{M}_{c,b}$ ($c \in \{1,2,\ldots,n\}$, $b \in \{0,1\}$). The state s_{end} is a common sink state, which is never left and obtains no rewards. The initial state of $\mathcal{M}(\phi)$ is state s_0, from which on action $\perp$ the initial state $[x_k = a]$ of each of the $\mathcal{A}_{k,a}$ is reachable with equal probability, i.e. with probability $\frac{1}{2n}$. Notice that from each of these states the same observation $*$ is made. The overall structure of $\mathcal{M}(\phi)$ is sketched in Figure 16.

Consider a formula $\phi \in$ QBF with variables $x_1, x_2, \ldots, x_n$, and consider $\mathcal{M}(\phi)$. In the first step, randomly a state $[x_k = a]$ is entered, and from the observation $*$ it cannot be concluded which one it is. In the next n steps, an assignment to the variables $x_1, x_2, \ldots, x_n$ is fixed. The observations made during these steps is the sequence of assignments made to each of the variables $x_1, \ldots, x_n$. Up to this point, each sequence of observations corresponding to an assignment (i.e. observations on trajectories with probability > 0 which do not reach the trap state s_{end}) appears for the same number of trajectories, namely for n many. Consequently, there are $2n \cdot 2^m$ trajectories – for m being the number of universally quantified variables of ϕ – which reach any of the initial states of $\mathcal{C}_{k,a}$ and on which a reward not equal 0 will be obtained. Each of these trajectories has equal probability. Because $\phi \in$ QBF, there exists a policy π such that all these trajectories represent assignments which satisfy ϕ. Now, assume that π is a policy, which is *consistent* with the observations from the n steps during the second stage – i.e. whenever it is "asked" to give an assignment to a variable, it does this according to the observations during the second stage and therefore it assigns the same value to every appearance of a variable in $\mathcal{C}_{k,a}$. Then every trajectory eventually gets reward 1. Summarized, if $\phi \in$ QBF, then there exists a *consistent* policy π under which $\mathcal{M}(\phi)$ has performance > 0.

If $\phi \notin$ QBF, then for every policy there is at least one trajectory that does not represent a satisfying assignment. If π is consistent, then such a trajectory obtains reward $-2n \cdot 2^m$. Because $\mathcal{M}(\phi)$ has $2n \cdot 2^m$ trajectories which get reward other than 0, it follows that the performance of π is at most $\frac{(2n \cdot 2^m - 1) - 2n \cdot 2^m}{2n \cdot 2^m} = \frac{-1}{2n \cdot 2^m}$, and that is < 0.

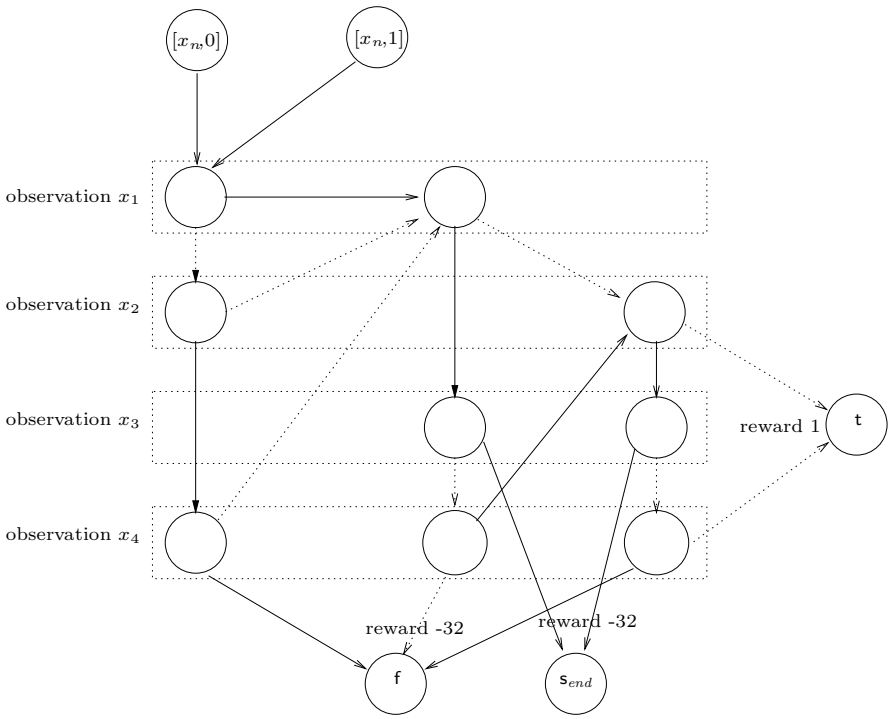


Figure 15: The third stage of $\mathcal{M}(\phi)$: $\mathcal{C}_{3,0}$ evaluating $(x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$.

It remains to consider the case that π is not a consistent policy. Hence, there is a state where x_k is observed, and π chooses an assignment a to x_k which differs from the assignment given to x_k on the $k+1$st step of the history. Then, on the trajectory through $\mathcal{C}_{k,1-b}$ a reward of $-2n \cdot 2^m$ is obtained. As above, the performance of π turns out to be negative.

Concluding, we have that there exists a policy under which $\mathcal{M}(\phi)$ has performance > 0 if and only if there exists a *consistent* policy under which $\mathcal{M}(\phi)$ has performance ≥ 0 if and only if $\phi \in$ QBF. The transformation of ϕ to $\mathcal{M}(\phi)$ can be performed in polynomial time. Hence, the **PSPACE**-hardness of the history-dependent policy existence problem is proven. □

The above theorem is in stark contrast to the infinite-horizon case, where the problem is simply uncomputable (Madani, Hanks, and Condon 1999). Problems in **PSPACE** are certainly computable, albeit potentially slowly: we have no evidence yet that there are **PSPACE**-complete problems with deterministic time complexity significantly below exponential time! Thus, limiting the horizon reduces an uncomputable problem to an intractable one. However, it does leave open the hope of reasonable heuristics.

For concise policies, the evaluation problem is in **PP**. Notice that other than for stationary, time-dependent, and history-dependent policies, a concise policy for a given $\mathcal{M}$ may have an arbitrary representation size. But we are interested in policies which can be represented in size polynomial in $\mathcal{M}$. For concise policies, we need to fix the degree of that polynomial in order to make it decidable within any nondeterministic and probabilistic polynomial-time bounds. Therefore, we say that the concise policy existence problem for POMDPs is in the complexity class $\mathcal{C}$, if there exists a constant $c > 0$, such that the c-concise policy existence problem for POMDPs is in $\mathcal{C}$. Given $\mathcal{M}$, a c-concise policy can be guessed in time $|\mathcal{M}|^c$. Hence, the guess-and-check approach

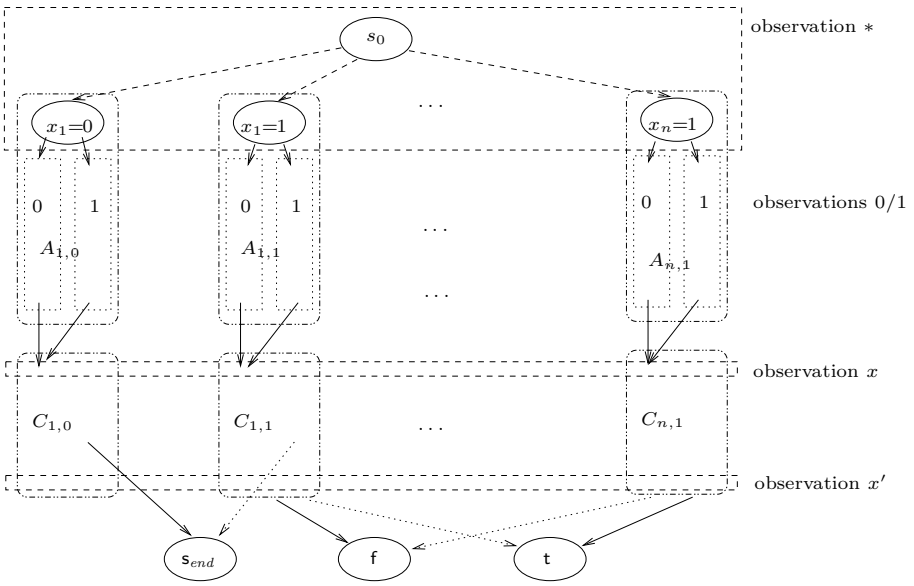


Figure 16: A sketch of $\mathcal{M}(\phi)$.

---

yields a straightforward upper bound of $\mathbf{NP}^{\mathbf{PP}}$ for the concise policy existence problem. In fact, this upper bound turns out to be the lower bound for the problem, too.

Theorem 5.10 [2] *The concise policy existence problem for POMDPs is* $\mathbf{NP}^{\mathbf{PP}}$*-complete.*

Proof. That the problem is in $\mathbf{NP}^{\mathbf{PP}}$ follows from the fact that a concise policy can be guessed in polynomial time and checked in **PP** according to Lemma 4.8.

To show $\mathbf{NP}^{\mathbf{PP}}$-hardness, we give a reduction from the $\mathbf{NP}^{\mathbf{PP}}$-complete problem EMAJSAT. This reduction is essentially the same as in the proof of Theorem 5.9. An instance of EMAJSAT is a pair (ϕ, i) consisting of a Boolean formula ϕ with variables $x_1, \ldots, x_n$ and a number $1 \leq i \leq n$. The pair (ϕ, i) is in EMAJSAT if and only if there is an assignment to the first i variables $x_1, \ldots, x_i$ such that the majority of assignments to the remaining $n - i$ variables $x_{i+1}, \ldots, x_n$ satisfies ϕ.

Given (ϕ, i), the POMDP $\mathcal{M}(\phi, i) = \mathcal{M}(\exists x_1 \cdots \exists x_i \forall x_{i+1} \cdots \forall x_n \phi)$ is constructed according to the construction in the proof of Theorem 5.9. If in the third stage the state t is reached, reward 1 is obtained, but if state f is reached, reward -1 is reached.

Consider any consistent policy π for $\mathcal{M}(\phi, i)$. Then, since each trajectory standing for a satisfying assignment gets reward 1 and each trajectory standing for an unsatisfying assignment gets reward -1, $\mathcal{M}(\phi, i)$ under this policy π has performance > 0 iff π assigned values to $x_1, \ldots, x_i$ such that the majority of assignments to the remaining variables $x_{i+1}, \ldots, x_n$ satisfies ϕ iff π "proves" that $(\phi, i) \in$ EMAJSAT. This means, there exists a consistent policy π under which $\mathcal{M}(\phi, i)$ has performance > 0 if and only if $(\phi, i) \in$ EMAJSAT.

Non-consistent policies will have negative perfromance.

Concluding, we have that there exists a policy under which $\mathcal{M}(\phi, i)$ has performance > 0 if and only if there exists a consistent policy under which $\mathcal{M}(\phi, i)$ has performance > 0 if and only if $(\phi, i) \in$ EMAJSAT. The transformation of (ϕ, i) to $\mathcal{M}(\phi, i)$ can be performed in polynomial time. Every consistent policy can be represented by a small circuit, because it consists of a simple table giving the assignments to the first i variables and a selector function when asked for assignments

in the second phase. Hence, the $\mathbf{NP}^{\mathbf{PP}}$-hardness of the concise policy existence problem is proven. □

5.4 Non-Approximability for Short-Term Policies

We have already seen that the policy existence problem is computationally intractable for most variations of POMDPs. For instance, we showed that the stationary policy existence problems for POMDPs with or without negative rewards are **NP**-complete. Computing an optimal policy is at least as hard as computing the value of a POMDP, what again is at least as hard as deciding the policy existence problem. Instead of asking for an optimal policy, we might wish to compute a policy that has a performance that is a large fraction of the value, or only approximate the value.

A polynomial-time algorithm computing a nearly optimal policy or value is called an *ε-approximation* (for $0 < \varepsilon \leq 1$), where ε indicates the quality of the approximation in the following way. Let us consider the value problem first. Let A be a polynomial-time algorithm which on input a POMDP $\mathcal{M}$ computes a value $v(\mathcal{M})$. The algorithm A is called an *ε-approximation* for the value problem, if for every POMDP $\mathcal{M}$,

$$v(\mathcal{M}) > (1 - \varepsilon) \cdot val_{\alpha,s}(\mathcal{M}).$$

For the optimal policy problem, we consider polynomial-time algorithms A which for every POMDP $\mathcal{M}$ compute a policy $\pi_\alpha^{\mathcal{M}}$. Then, algorithm A is called an *ε-approximation* for the optimal policy problem, if for every POMDP $\mathcal{M}$,

$$perf_s(\mathcal{M}, \pi_\alpha^{\mathcal{M}}) > (1 - \varepsilon) \cdot val_{\alpha,s}(\mathcal{M}).$$

(See e.g. (Papadimitriou 1994) for more detailed definitions.) Approximability distinguishes **NP**-complete problems: there are problems which are ε-approximable for all ε, for certain ε, or for no ε (unless $\mathbf{P} = \mathbf{NP}$).

Notice that the policy existence problem for POMDPs with negative and nonnegative rewards is not suited for approximation. If a policy with positive performance exists, than every approximation algorithm yields such a policy, because a policy with performance 0 or smaller cannot approximate a policy with positive performance. Hence, any approximation solves an **NP**-complete problem. Therefore, we consider POMDPs with nonnegative rewards only. The first question is whether an optimal stationary policy can be ε-approximated for POMDPs with nonnegative rewards. In Corollary 5.6 it was shown that the related decision problem is **NP**-complete.

Theorem 5.11 *(see also* [4] *) Let $0 < \varepsilon \leq 1$. The stationary optimal policy problem for POMDPs with positive performance under every policy is ε-approximable if and only if $\mathbf{P} = \mathbf{NP}$.*

Proof. The stationary value of a POMDP can be calculated in polynomial time by a binary search method using an oracle solving the stationary policy existence problem for POMDPs. Knowing the value, one can try out to fix an action for an observation. If the modified POMDP still achieves the value calculated before, one can continue with the next observation, until a stationary policy is found which has the optimal performance. This algorithm runs in polynomial time using an oracle solving the stationary policy existence problem for POMDPs. Since the oracle is in **NP** by Theorem 5.5, the algorithm runs in polynomial time, if $\mathbf{P} = \mathbf{NP}$.

Now, assume that A is a polynomial-time algorithm that ε-approximates the optimal stationary policy for some ε with $0 < \varepsilon \leq 1$. We show that this implies that $\mathbf{P} = \mathbf{NP}$ by showing how to solve the **NP**-complete problem 3SAT. As in the proof of Theorem 5.5, given a formula ϕ being

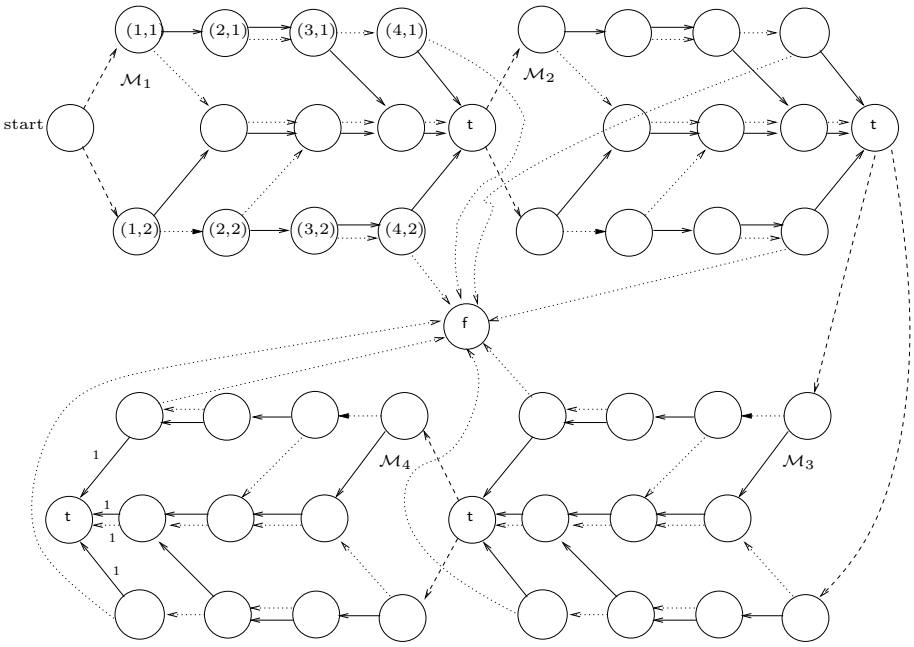


Figure 17: An example unobservable POMDP for $\phi = (\neg x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_4)$

---

an instance of 3SAT, we construct a POMDP $\mathcal{M}(\phi)$. We only change the reward function of the POMDP constructed in the proof of Theorem 5.5, in order to make it a POMDP with positive performances. Now, reward 1 is obtained if state f is reached, and reward $\lceil 1 + \frac{1}{\varepsilon} \rceil$ is obtained if state t is reached. Hence ϕ is satisfiable if and only if $\mathcal{M}(\phi)$ has value $\lceil 1 + \frac{1}{\varepsilon} \rceil$.

Assume that policy π is the output of the approximation algorithm A. If ϕ is satisfiable, then $perf_s(\mathcal{M}(\phi), \pi) \geq \varepsilon \cdot (1 + \frac{1}{\varepsilon}) = \varepsilon + 1 > 1$. Because the performance of every policy for $\mathcal{M}(\phi)$ is either 1, if ϕ is not satisfiable, or $\lceil 1 + \frac{1}{\varepsilon} \rceil$, if ϕ is satisfiable, it follows that π has performance > 1 if and only if ϕ is satisfiable. So, in order to decide $\phi \in$ 3SAT one can construct $\mathcal{M}(\phi)$, run the approximation algorithm A on it, take its output π and calculate $perf_s(\mathcal{M}(\phi), \pi)$. Dependent on that output it can be decided whether ϕ is in 3SAT. All these steps are polynomial-time bounded computations. It follows that 3SAT is in $\mathbf{P}$, and hence $\mathbf{P} = \mathbf{NP}$. □

Using the same proof technique as above, we can show that the value problem is non-approximable, too.

Corollary 5.12 *Let $0 < \varepsilon \leq 1$. The stationary value problem for POMDPs with positive performance under every policy is ε-approximable if and only if $\mathbf{P} = \mathbf{NP}$.*

For POMDPs with nonnegative rewards, the time-dependent policy existence problem is **NL**-complete (Theorem 5.8), what seems to be much easier than the problem for stationary policies. Nevertheless, its approximability is tied to the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ question.

Theorem 5.13 *(see [4]) Let $0 < \varepsilon \leq 1$. The time-dependent optimal policy problem for unobservable POMDPs with positive performances is ε-approximable if and only if $\mathbf{P} = \mathbf{NP}$.*

Proof. We give a reduction from 3SAT with the following properties. For a formula ϕ we show how to construct an unobservable POMDP $\mathcal{M}_\varepsilon(\phi)$ with value 1 if ϕ is satisfiable, and with value

$< (1-\varepsilon)$ if ϕ is not satisfiable. Therefore, an ε-approximation could be used to distinguish between satisfiable and unsatisfiable formulas in polynomial time.

Given a formula ϕ, We first show how to construct an unobservable $\mathcal{M}(\phi)$ from which $\mathcal{M}_\varepsilon(\phi)$ will be constructed. The construction of $\mathcal{M}(\phi)$ is the same as in the proof of Theorem 5.2, but with reward 0 (instead of $-m$) for clauses that are not satisfied. Notice that $\mathcal{M}(\phi)$ has value 1, if ϕ is satisfiable. If ϕ is not satisfiable, then all but one of the m clauses of ϕ may be simultaneously satisfiable and hence the value of $\mathcal{M}(\phi)$ may be $1-\frac{1}{m}$. This value comes arbitrarily close to 1 (for large m). Now, construct $\mathcal{M}_\varepsilon(\phi)$ from m^2 copies $\mathcal{M}_1, \ldots, \mathcal{M}_{m^2}$ of $\mathcal{M}_\phi$, such that the initial state of $\mathcal{M}_\varepsilon(\phi)$ is the initial state of $\mathcal{M}_1$, the initial state of $\mathcal{M}_{i+1}$ is the state t of $\mathcal{M}_i$, and reward 1 is gained if the state t of $\mathcal{M}_{m^2}$ is reached. The error states f of all the $\mathcal{M}_i$s are identified as a unique sink state f.

To illustrate the construction, in Figure 17 we give an example POMDP consisting of a chain of 4 copies of $\mathcal{M}(\phi)$ obtained for the formula $\phi = (\neg x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_4)$. The dashed arrows indicate a transition with probability $\frac{1}{2}$. The dotted (resp. solid) arrows are probability 1 transitions on action 0 (resp. 1).

If ϕ is satisfiable, then a time-dependent policy simulating m^2 repetitions of any satisfying assignment has performance 1. If ϕ is not satisfiable, then under any assignment at least one of the m clauses of ϕ is not satisfied. Hence, the probability that under any time-dependent policy the final state t of $\mathcal{M}(\phi)$ is reached is at most $1-\frac{1}{m}$. Consequently, the probability that the final state of $\mathcal{M}_\varepsilon(\phi)$ is reached is at most $(1-\frac{1}{m})^{m^2} \leq e^{-m}$. This probability equals the value of $\mathcal{M}_\varepsilon(\phi)$. Since for large enough m it holds that $e^{-m} < (1-\varepsilon)$. Remember that m is the number of clauses of ϕ, and ϕ is in 3CNF. Because there is only a finite number of non-isomorphic 3CNF formulas with at most m clauses, an ε-approximation algorithm would directly yield a polynomial-time algorithm for all but finitely many instances of 3SAT. □

Because unobservability is a special case of partial observability, we get the same non-approximability result for POMDPs and also for unrestricted rewards.

Corollary 5.14 *Let $0 < \varepsilon \leq 1$. The optimal time-dependent policy problem for POMDPs is ε-approximable if and only if* $\mathbf{P} = \mathbf{NP}$.

Using the same technique as in the proof of Theorem 5.13 we obtain the non-approximability of the value problem.

Corollary 5.15 *Let $0 < \varepsilon \leq 1$. The time-dependent value problem for POMDPs with positive performances is ε-approximable if and only if* $\mathbf{P} = \mathbf{NP}$.

A short-term history-dependent policy for a POMDP $\mathcal{M}$ may not be representable in space polynomial in $|\mathcal{M}|$. Therefore, we cannot expect that a polynomial-time algorithm on input $\mathcal{M}$ outputs a history-dependent policy, and we restrict consideration to polynomial-time algorithms that approximate the history-dependent value of a POMDP with nonnegative rewards. Notice that the related policy existence problem is only **NL**-complete (Theorem 5.8). Since **NL** is properly included in **PSPACE**, the approximability is strictly harder.

Theorem 5.16 *Let $0 < \varepsilon \leq 1$. The history-dependent value problem for POMDPs with nonnegative rewards is ε-approximable if and only if* $\mathbf{P} = \mathbf{PSPACE}$.

Proof. The history-dependent value of a POMDP $\mathcal{M}$ can be calculated using binary search over the history-dependent policy existence problem. The number of bits to be calculated is polynomial

in the size of $\mathcal{M}$. Therefore, by Theorem 5.9, this calculation can be performed in polynomial time using a **PSPACE** oracle. If **P** = **PSPACE**, it follows that the history-dependent value of a POMDP $\mathcal{M}$ can *exactly* be calculated in polynomial time.

The set QBF of true quantified Boolean formulas is **PSPACE**-complete. It can be interpreted as a two-player game: player 1 sets the existentially quantified variables, and player 2 sets the universally quantified variables. The goal of player 1 is to have a satisfying assignment to the formula after the alternating choices, and player 2 has the opposite goal. A formula is in QBF if and only if player 1 has a winning strategy. This means player 1 has a response to every choice of player 2 so that in the end the formula will be satisfied.

The version where player 2 makes random choices and player 1's goal is to win with probability $> \frac{1}{2}$ corresponds to SSAT (*stochastic satisfiability*), which is also **PSPACE** complete. The instances of SSAT are formulas which are quantified alternatingly with existential quantifiers $\exists$ and random quantifiers R. The meaning of the random quantifier R is that an assignment to the respective variable is chosen randomly. A stochastic Boolean formula

$$\Phi = \exists x_1 R x_2 \exists x_3 R x_4 \ldots \phi$$

is in SSAT if and only if

$$\text{there exists } b_1 \text{ for random } b_2 \text{ exists } b_3 \text{ for random } x_4 \ldots$$
$$Prob[\phi(b_1, \ldots, b_n) \text{ is true}] > \tfrac{1}{2}.$$

From the proof of **IP** = **PSPACE** by Shamir (1992) it follows that every instance x of a **PSPACE** set A can be polynomial-time transformed to a *bounded error* stochastic Boolean formula $\exists x_1 R x_2 \ldots \phi$ such that

- if $x \in A$, then
  $\exists b_1$ for random $b_2 \ldots Prob[\phi(b_1, \ldots, b_n) \text{ is true}] > (1 - 2^{-c})$, and
- if $x \notin A$, then $\forall b_1$ for random $b_2 \ldots Prob[\phi(b_1, \ldots, b_n) \text{ is true}] < 2^{-c}$

where $c \geq 1$ is an arbitrary constant. This means that player 1 either has a strategy under which he wins with very high probability, or the probability of winning (under any strategy) is very small. We show how to transform a stochastic Boolean formula Φ into a POMDP whose history-dependent value is close to 1, if player 1 has a winning strategy, and far from 1, if player 2 wins. The construction of this POMDP $\mathcal{M}(\Phi)$ from a stochastic Boolean formula Φ is the same as in the proof of Theorem 5.9, but with the negative rewards replaced by reward 0. If $\Phi \in$ SSAT, then $\mathcal{M}(\Phi)$ has value $> 1 - 2^{-c}$. But, this change of the rewards has as consequence, that a non-consistent policy – i.e. one that "cheats" and gives a different assignment to a variable during the third (checking) stage than it did during the second stage – is not anymore "punished" so hard, that it has a lower performance than a consistent policy. Therefore, we cannot conclude that $\mathcal{M}(\Phi)$ has value $< 2^{-c}$, if $\Phi \notin$ SSAT. Anyway, a non-consistent policy is trapped on cheating on at least one assignment to one of the n variables of the formula, and therefore at least one of the $2n$ parallel processes in stage 2 of $\mathcal{M}(\Phi)$ yields reward 0 on all trajectories. Hence, the performance of a non-consistent policy is at most $1 - \frac{1}{2n}$. Now, we do an amplification similar as in the proof of Theorem 5.13. We run $\mathcal{M}(\Phi)$ for $(2n)^2$ times, where rewards other than 0 are obtained only in the last run, and only on those trajectories that were never trapped at cheating and that satisfy the formula (even with untrapped cheating). Trajectories on which the policy is trapped at cheating at least once have reward 0. Let $\widehat{\mathcal{M}(\Phi)}$ be this POMDP. Hence, the performance of $\widehat{\mathcal{M}(\Phi)}$ under non-consistent policies is at most $(1 - \frac{1}{2n})^{(2n)^2}$. Because for all constants ε and c with $0 \leq \varepsilon < 1$ and $c \geq 1$ it holds

for almost all n that $(1-\frac{1}{2n})^{(2n)^2} \leq e^{-2n} < (1-\varepsilon)\cdot(1-2^{-c})$, it follows that the performance of a non-consistent policy does not ε-approximate the value of $\widehat{\mathcal{M}(\Phi)}$. Now, we choose c in a way that $2^{-c} < (1-\varepsilon)\cdot(1-2^{-c})$. Then it follows that

- if $\Phi \in$ SSAT, then $\widehat{\mathcal{M}(\Phi)}$ has value $> 1-2^{-c}$, and
- if $\Phi \notin$ SSAT, then $\widehat{\mathcal{M}(\Phi)}$ has value $< (1-\varepsilon)\cdot(1-2^{-c})$.

Hence, an ε-approximation of the value of $\widehat{\mathcal{M}(\Phi)}$ already decides whether Φ is in SSAT.

Concluding, suppose that the history-dependent value of POMDPs has a polynomial-time ε-approximation. Choose c dependent on ε as described. Let A be any set in **PSPACE**. There exists a polynomial-time function f which maps every instance x of A to a bounded error stochastic formula $f(x) = \Phi_x$ with error 2^{-c} and reduces A to SSAT. Transform Φ_x into the POMDP $\widehat{\mathcal{M}(\Phi_x)}$. Using the ε-approximate value of $\widehat{\mathcal{M}(\Phi_x)}$, one can decide $\Phi_x \in$ SSAT and hence $x \in A$ in polynomial time. This shows that A is in **P**, and consequently **P** = **PSPACE**. □

The results of this Section show that the complexity of the approximability of value problems for POMDPs with *nonnegative* rewards is equal to the complexity of the policy existence problems for POMDPs with both *negative and nonnegative* rewards. In all the proofs of this Section we used some kinds of amplification techniques, that guarantee the value of a POMDP to be either very large or very small. Such techniques are yet not known for problems in **PP** and for $\mathbf{NP}^{\mathbf{PP}}$. Therefore, we are yet not able to prove a non-approximability result for concise policies.

6 Short-Term Policy Existence for Compressed POMDPs

Compressed POMDPs are represented by circuits. The transition probability function is represented by a circuit. This circuit gets as input a triple (s, a, s'), the binary encoding of states s and s', and the binary encoding of action a. It outputs all the bits of the transition probability $t(s, a, s')$. The observation function is represented by a circuit, that gets a binary encoding of a state as input and outputs a binary encoding of an observation. The reward function is represented by a circuit, that gets a binary encoding of a state and an action as input and outputs the reward in binary. This means, that the number of states, actions, and observations may be up to exponential in the size of the circuits. On the other hand, the transition probabilities and rewards are not larger than with flat representations (compared to the representation size), because there they were represented in binary, too. This makes compressed representations different from succinct representations. With succinct representations, we can expect a full exponential jump in the complexity compared to flat representations. This means, short-term problems with flat representation that are in **P**, **NP**, or **PSPACE** transform to long-term problems with succinct representations that are in **EXP**, **NEXP**, or **EXPSPACE**. In this section, we investigate short-term problems with compressed representations. In a sense, this is only "half" an exponential jump. Such jumps are trading time resources by space resources, for example from **P** to **PSPACE**, or trading space resources by exponentially larger time resources, for example from **PL** to **PP**. Surprisingly, everything goes. There are problems that make a full exponential jump, others that make a half exponential jumps, and also some whose complexity does not increase at all.

6.1 Fully-Observable POMDPs

For (fully-observable) flat MDPs, we have **P**-completeness for the time-dependent and for the history-dependent existence problems (Theorem 5.4), and we have **P**-hardness and containment in

NP for the stationary problem (Theorem 5.3). The role of **P** is taken by **PSPACE**, what is a "half exponential" jump.

Lemma 6.1 [3] *The stationary, time-dependent and history-dependent policy existence problems for compressed MDPs are* **PSPACE***-hard.*

Proof. To prove hardness, we show a polynomial-time reduction from QBF, the validity problem for quantified Boolean formulas.

From a formula Φ with n quantified variables, we construct a fully-observable MDP with $2^{n+1}-1$ states, where every state represents an assignment of Boolean values to the first i variables ($0 \leq i \leq n$) of Φ. Transitions from state s can reach the two states representing assignments that extend s by assigning a value to the next unassigned variable. If this variable is bound by an existential quantifier, then the action taken in s assigns a value to that variable; otherwise the transition is random and independent of the action. Reward 1 is gained for every action after a state representing a satisfying assignment for the formula is reached. If a state representing an unsatisfying assignment is reached, reward $-(2^n)$ is gained. Then the maximal performance of this MDP is positive if and only if the formula is true. A compressed representation of the MDP can be computed in time polynomial in the size of the formula Φ.

Since every state except the last one appears at most once in every trajectory, this construction proves the same lower bound for any type of policy. □

Lemma 6.2 [3] *The history-dependent policy existence problem for compressed MDPs is in* **PSPACE***.*

Proof. In **PSPACE**, an entire history-dependent policy cannot be specified for a horizon n equal to the size of the input. However, a policy can be guessed stepwise. Therefore, we show that this problem is in **NPSPACE**, what equals **PSPACE**.

The set of possible histories to consider forms a tree of depth n. A policy can be specified by labeling the edges of that tree with actions. Such a policy can be guessed one branch at a time; in order to maintain consistency, one need only keep track of the current branch.

To evaluate a policy, for each branch through the policy tree, one must compute the value of each node under that policy. The probability of each transition can be represented with n bits; the product of n such requires n^2 bits. Rewards require at most n bits, and are accumulated (potentially) at each transition. The total reward for a given trajectory, therefore, requires at most n^3 bits.

There are at most 2^n states, so there are at most 2^{n^2} trajectories. The value of each is bounded by 2^{n^3}, so the sum is bounded by $2^{n^2+n^3}$, and thus can be represented by polynomially many bits. □

As in the case for flat MDPs, one can argue that the optimal action chosen when state s is reached in step i does not depend on how state s was reached (i.e. on the history of the process). Hence, the maximal performance of a compressed MDP under history-dependent policies equals its maximal performance under time-dependent policies. Therefore, the time-dependent policy existence problem for this type of MDP has the same complexity as the history-dependent problem.

Lemma 6.3 [3] *The time-dependent policy existence problem for compressed MDPs is in* **PSPACE***.*

Combining Lemma 6.2 and Lemma 6.3 with Lemma 6.1, we show that the complexity of policy existence problems makes a jump from polynomial time to polynomial space if we consider compressed MDPs instead of flat MDPs. One could argue that this is a good indication that **P** is different from **PSPACE**. However, this remains an open question.

Theorem 6.4 [3] *The time-dependent and history-dependent policy existence problems for compressed MDPs are* **PSPACE***-complete.*

Similarly to flat MDPs, the exact complexity of the stationary policy existence problem is open. Its straightforward upper bound is **NEXP**: given a compressed MDP $\mathcal{M}$, write down its flat representation (that takes exponential time), guess a stationary policy (nondeterministic exponential time), and calculate its performance for $|\mathcal{M}|$ steps (exponential time). But this is quite far away from the lower bound **PSPACE** (Theorem 6.1).

6.2 POMDPs

For compressed POMDPs, we could expect to obtain similar complexity jumps as from flat to compressed MDPs. Surprisingly, the complexities of the different policy existence problems relate totally differently. The history-dependent existence problems do not alter its complexity from flat to compressed representation. The concise policy existence problem becomes the easiest of all. And the history-dependent policy existence problem remains easier than the stationary and the time-dependent existence problems, that make a full (!) exponential complexity jump.

Theorem 6.5 [3] *The stationary policy existence problem for compressed POMDPs is* **NEXP***-complete.*

Proof. Membership in **NEXP** follows from the standard guess-and-check approach, as for compressed MDPs. To show **NEXP**-hardness, we sketch a reduction from the succinct version of 3Sat, shown to be **NEXP**-complete in (Papadimitriou and Yannakakis 1986), to the stationary existence problem. Instances for succinct 3Sat are encodings of Boolean formulas in conjunctive normal form, in which all clauses consist of exactly three literals, and each literal appears at most three times. Hence, if the formula has n variables, it has $O(n)$ clauses. The encoding of the formula is a circuit which on input $i; k$ outputs the ith literal in the kth clause. The reduction is similar to that of Theorem 5.2, where a formula was transformed into a POMDP which "checks" all clauses of the formula in parallel. Here, we put all states which stand for appearances of the same variable into one observation class and leave out considerations of variables in a clause where they do not appear. Actually, this is the place where we use partially observability instead of unobservability as in the proof of Theorem 5.2. This drastically speeds up the process from needing exponentially many steps to a constant number of steps.

Let m be the number of clauses of the formula. Then each of the m trajectories consists of the initial state, (at most) three states for the literals in one clause and a final sink. Each stationary policy corresponds to an assignment to the variables. If the assignment is satisfying, each trajectory will yield reward 1. If the assignment is not satisfying, then at least one trajectory corresponds to an unsatisfied clause and yields reward $-m$. Because there are at most $3m$ variables, the number $-m$ can be compressed represented.

Figure 18 sketches the construction for our standard example. □

The time-dependent policy existence problem for compressed POMDPs has the same complexity. Interestingly, as in the flat case we need more random transitions to show hardness for the time-dependent existence problem than for the stationary one.

Theorem 6.6 [3] *The time-dependent policy existence problem for compressed POMDPs is* **NEXP***-complete.*

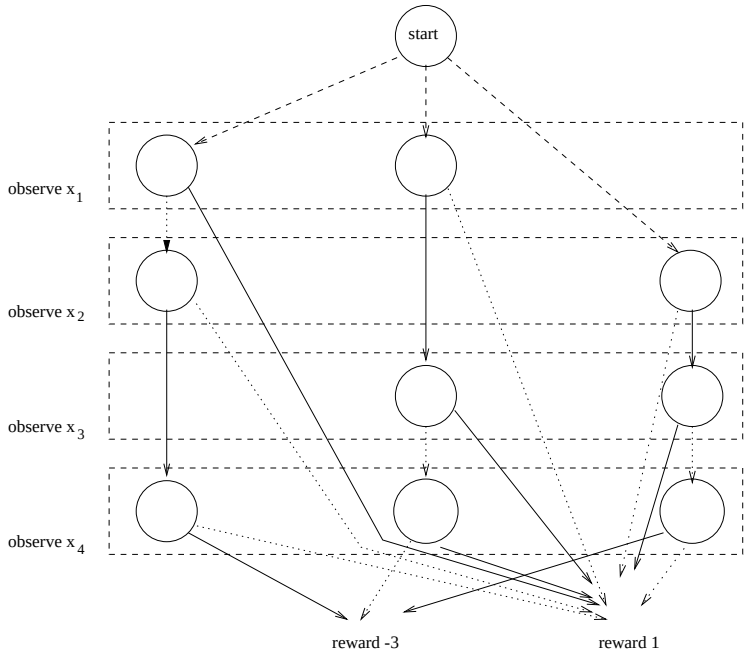


Figure 18: $\mathcal{M}(\phi)$ for $\phi = (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$.

Proof. Containment in **NEXP** follows from the standard guess-and-check approach. To show hardness for **NEXP**, we give a reduction from succinct 3SAT. Similar to the construction in the proof of Theorem 5.9, we use three stages: the first stage randomly chooses a variable, the second stage lets the policy fix an assignment to it, and the third stage checks whether under this assignment the policy provides a satisfying assignment to the whole formula. The third stage is similar to the construction in the proof of Theorem 6.5. Let $\mathcal{M}_{(j,b)}$ be the POMDP constructed as in the proof of Theorem 6.5, but whenever observation x_j is made (i.e. the process expects an assignment to variable x_j) then reward -2^{n+m} is obtained if action b is *not* chosen. Let $s_{(j,b)}$ be the initial state of $\mathcal{M}_{(j,b)}$. The new process we construct has a new initial state. Let l be the number of variables in the formula. From the initial state with equal probability the new states s_i (for $1 \leq i \leq n$) are reachable. State s_i has observation x_i, and the meaning of s_i is to fix an assignment to x_i. On action $a \in \{0, 1\}$, the process goes deterministically from state s_i to state $s_{(i,a)}$. Notice that after the next transition the observations are in $\{x_1, \ldots, x_l\}$, and henceforth the time-dependent policy cannot remember which variable it assigned on the first step. Assume that at some step the policy acts differently on observation x_i than in the first step, where it chose action b. Then it will obtain such a big negative reward in the subprocess $\mathcal{M}_{(i,b)}$ that the expected reward of the whole process will be negative. Therefore, the only policy that obtains a positive expected reward is the one that behaves consistently, i.e. which chooses the same action whenever the same observation is made. As in the above proof, it now follows that a time-dependent policy with positive expected reward exists if and only if the formula is satisfiable.

Figure 19 sketches the construction for our standard example. □

Full or partial observability does not affect the complexity of the history-dependent existence problems.

Theorem 6.7 *The concise policy existence problem for compressed POMDPs is* $\mathbf{NP}^{\mathbf{PP}}$*-complete.*

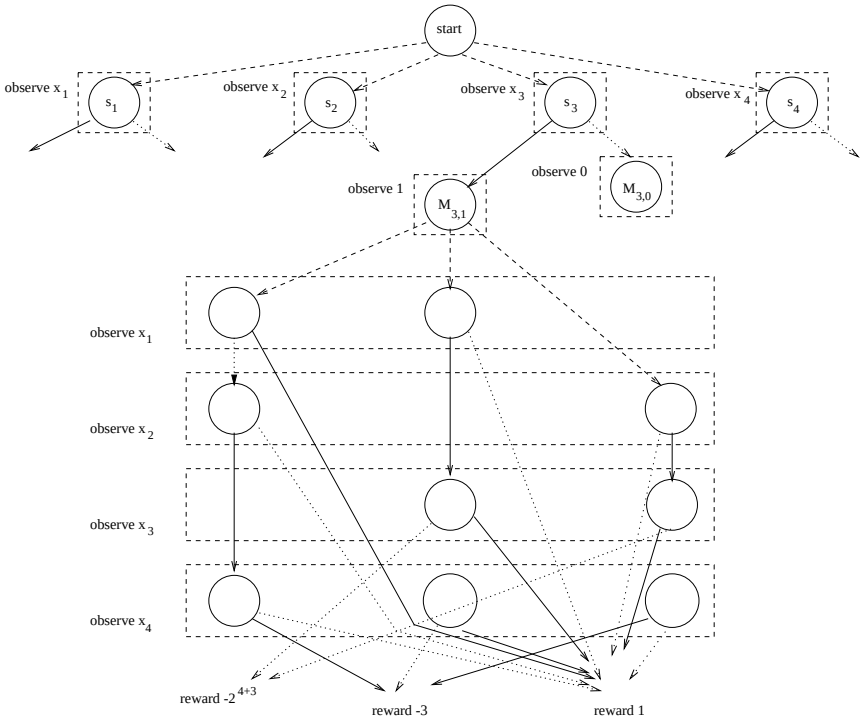


Figure 19: $\mathcal{M}(\phi)$ for $\phi = (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$.

Proof. That the problem is in $\mathbf{NP^{PP}}$ follows from the fact that a concise policy can be guessed in polynomial time and checked in **PP** according to Theorem 4.14. Hardness for $\mathbf{NP^{PP}}$ follows from $\mathbf{NP^{PP}}$-hardness of the flat case (Theorem 4.9). □

Theorem 6.8 [3] *The history-dependent policy existence problem for compressed POMDPs is* **PSPACE***-complete.*

Proof. The argument that the problem is in **PSPACE** is the same as in the proof of Theorem 5.9. Hardness for **PSPACE** follows from Lemma 6.1. □

For POMDPs with nonnegative rewards, the policy existence problems reduce to searching for short paths in succinctly represented graphs – e.g. an accepting path in the computation graph of an **NP** machine – and vice versa, similar to the proof of Theorem 5.8. This yields straightforwardly **NP**-completeness.

Theorem 6.9 [3] *The stationary, time-dependent and history-dependent policy existence problems for compressed POMDPs with nonnegative rewards all are* **NP***-complete.*

6.3 Unobservable POMDPs

For flat POMDPs, the complexity of policy existence is equal for partial observability and unobservability. We show that observability influences the complexity of compressed POMDPs. The policy existence problems for compressed UMDPs are simpler than for compressed MDPs. Later, we will show that the number of possible actions is important.

Theorem 6.10 [3] *The stationary policy existence problem for compressed UMDPs is complete for* $\mathbf{NP}^{\mathbf{PP}}$.

Proof. To see that this problem is in $\mathbf{NP}^{\mathbf{PP}}$, remember that the corresponding policy evaluation problem is **PP**-complete, by Theorem 4.12. For the existence question, one can guess a (polynomial-sized) policy, and verify that it has performance > 0 by consulting a **PP** oracle.

To show $\mathbf{NP}^{\mathbf{PP}}$-hardness, one needs that $\mathbf{NP}^{\mathbf{PP}}$ equals the $\leq_m^{np}$ closure of **PP** (Torán 1991). Hence, $\mathbf{NP}^{\mathbf{PP}}$can be seen as the closure of **PP** under polynomial-time disjunctive reducibility with an exponential number of queries. Each of these queries is polynomial-time computable from its index in the list of queries.

Let $A \in \mathbf{NP}^{\mathbf{PP}}$. By the above observations and Theorem 4.12, there is a polynomial-time computable two-parameter function f and a polynomial p, such that for every x, $x \in A$ if and only if there exists a y of length $p(|x|)$ such that $f(x, y)$ outputs an element $(\mathcal{M}, \pi)$ of the stationary policy evaluation problem for compressed UMDPs. We fix an x. For $f(x, y) = (\mathcal{M}, \pi)$, we let $\mathcal{M}_{x,y}$ be that UMDP obtained from $\mathcal{M}$ by hard-wiring policy π into it. I.e. $\mathcal{M}_{x,y}$ has the same states as $\mathcal{M}$, but whenever an observation b is made in $\mathcal{M}$, $\mathcal{M}_{x,y}$ on any action a behaves like $\mathcal{M}$ on action $\pi(b)$. Hence, we do not need any observations in $\mathcal{M}_{x,y}$, and $\mathcal{M}$ has positive performance under π if and only if $\mathcal{M}_{x,y}$ has positive performance after the given number of steps under all policies. After this given number of steps, $\mathcal{M}_{x,y}$ goes into a sink state in which it earns no further rewards. Now, we construct a new UMDP $\mathcal{M}_x$ from the union of all $\mathcal{M}_{x,y}$ and a new initial state. From its initial state, $\mathcal{M}_x$ on action a reaches the initial state of $\mathcal{M}_{x,a}$ with probability 1 and reward 0. Therefore, $\mathcal{M}_x$ has positive performance under some policy if and only if for some a, $\mathcal{M}_{x,a}$ has positive performance under constant policy a if and only if for some a, $f(x, a)$ is an element of the policy evaluation problem under consideration if and only if $x \in A$. Because f is polynomial time computable, and $\mathcal{M}_x$ has at most $2^{p(|x|)+1}$ states, it follows that $\mathcal{M}_x$ is a compressed UMDP. Thus the policy existence problem is hard for $\mathbf{NP}^{\mathbf{PP}}$. □

Note that only the action chosen in the first step determined which reward will be obtained. Therefore, we obtain the same hardness for time-dependent policies. Membership in $\mathbf{NP}^{\mathbf{PP}}$ follows from Theorem 4.12.

Theorem 6.11 [3] *The time-dependent policy existence problem for com- pressed UMDPs is* $\mathbf{NP}^{\mathbf{PP}}$*-complete.*

It turns out that the number of actions of the UMDP affects the complexity. In the proof of Theorem 6.11 we needed a number of actions exponential in the size of the UMDP. Let us restrict this number to be polynomial in the size of the UMDP. Those UMDPs will be called *UMDPs with few actions* and have lower complexity.

Theorem 6.12 [3] *The stationary policy existence problem for compressed UMDPs with few actions is* **PP***-complete.*

Proof. Hardness for **PP** follows from the hardness proof of Lemma 4.3. To show containment in **PP**, we use arguments similar to those in the proof of Lemma 4.2. It is convenient to use the class **GapP** , which is the class of functions that are the difference of two #**P** functions. We make use of the fact that $A \in \mathbf{PP}$ if and only if there exists a **GapP** function f such that for every x, $x \in A$ if and only if $f(x) > 0$ (see Fenner, Fortnow, and Kurtz (1994)). One can show that the function p from the proof of Lemma 4.2 is in **GapP**, because the respective matrix powering is in **GapP** (see the proofs of Lemmas 4.11 and 4.1), and **GapP** is closed under multiplication and

summation. Finally, **PP** is closed under polynomial-time disjunctive reducibility (Beigel, Reingold, and Spielman (1995)), which completes the proof. □

The complexity gap between the stationary and the time-dependent policy existence problems for compressed UMDPs with few actions is as big as that for flat POMDPs, but the difference no longer depends on the number of actions.

Theorem 6.13 [3] *The time-dependent policy existence problem for compressed UMDPs with few actions is* $\mathbf{NP}^{\mathbf{PP}}$*-complete.*

Proof. In the proof of Lemma 6.10, we consider a polynomial-time computable two-parameter function f and a polynomial p, such that for every x, $x \in A$ if and only if there exists a y of length $p(|x|)$ such that $f(x, y)$ outputs a positive instance $(\mathcal{M}, \pi)$ of the stationary policy evaluation problem for compressed UMDPs. The UMDP $\mathcal{M}_{x,y}$ has exactly one action, and the $\mathcal{M}_{x,y}$s are joined so that action y in step 1 transitions to $\mathcal{M}_{x,y}$.

We now modify that construction by allowing $p(|x|)$ steps to get to y, and 2 actions (0 and 1); a time-dependent policy specifies a sequence of 0s and 1s (which we call "y") for the first $p(|x|)$ steps, and then both actions have the same effect within $\mathcal{M}_{x,y}$.

Thus, $x \in A$ if and only if there is some y such that $f(x, y) = \mathcal{M}_{x,y}$ has positive expected value if and only if there is some time-dependent policy for $\mathcal{M}_x$ that has positive expected value. □

7 Long-Term Policies

A long-term horizon is exponential in the size of the POMDP description. One can argue that, for compressed or succinct representations, this is the natural analogue of a short-term horizon that is size of a flat representation. If an example of a short-horizon policy is emergency medical care, then a long-term horizon would be the ongoing policy of, say, a family practice doctor. Or, for another example, consider radioactive waste management: congressional policy may be short-term, but a reasonable and safe policy should be extremely long-term. (We won't even mention, for instance, U.S. funding policies for basic research.)

For long-term problems, compressed POMDPs are more interesting than flat POMDPs, because in the long-term case the number of states is exponential in the size of the input, so a trajectory does not need to visit a single state too often. Remember that we have two circuit representations of POMDPs. A *succinct POMDP* consists of circuits calculating the transition probability t, the observation function o, and the reward function r. Unlike compressed POMDPs, these circuits only have one output bit. They take the index of the function value's bit as additional input. For example, the state transition probability function t is represented by a circuit which on input (s, a, s', l) outputs the lth bit of $t(s, a, s')$. This allows for function values whose binary representations are exponentially long in the size of the function's circuit representation. This is far beyond tractability. It is easy to see that in going from the short-term flat to long-term succinct problems, the complexity of the corresponding decision problems increases at most exponentially. More importantly (and less obviously), there are many problems for which this exponential increase in complexity is inherent (see Balcázar, Lozano, and Torán (1992) for general conditions for such an increase). This also holds for succinctly represented POMDPs (see Mundhenk, Goldsmith, and Allender (1997)). The complexity of long-term problems for compressed POMDPs is not so straightforward, because the strength of compressed representations is intermediate between flat and succinct representations. Also – as shown in Theorem 2.2 – compressed representation is similar to other compact representations used in AI. Therefore, we concentrate on compressed represented

POMDPs here. Actually, we do not get better complexity results for compressed POMDPs than for succinct POMDPs. In a sense, this shows that such small probabilities, that can be succinctly but not compressed represented, have no impact on the structural hardness of the problems. What makes the problems hard is their running time, i.e. their horizon.

7.1 Policy Evaluation

As in the short-term problems for compressed POMDPs, specification of long-term policies for compressed POMDPs is not tractable. Moreover, because the horizon is exponential in the size of the POMDP's description, even time-dependent policies for UMDPs cannot be efficiently specified. Therefore, we consider the complexity of policy evaluation only for UMDPs under stationary policies.

Put succinctly, the techniques from Section 4.1 translate to compressed represented POMDPs.

The first result which we translate is Lemma 4.1, where we considered the problem of matrix powering for $m \times m$ matrices; here, we consider powering a succinctly-represented $2^m \times 2^m$ matrix. In space logarithmic in the size of the matrix, we can apply the algorithm posited in Lemma 4.1; this locates the problem in the function class #**PSPACE**.

Lemma 7.1 [3] *Let T be a $2^m \times 2^m$ matrix of nonnegative integers, each consisting of 2^m bits. Let T be represented by a Boolean circuit C with $3m$ input bits, such that $C(a, b, r)$ outputs the r-th bit of $T_{(a,b)}$. For $1 \leq i, j \leq 2^m$, and $0 \leq s \leq m$, the function mapping (C, s, i, j) to $(T^s)_{(i,j)}$ is in* #**PSPACE**.

As a consequence, we can prove the long-term policy evaluation problem for compressed UMDPs is exponentially more complex than for the flat ones.

Theorem 7.2 [3] *The long-term stationary policy evaluation problem for compressed UMDPs is* **PSPACE***-complete.*

Proof. In order to show that the problem is in **PSPACE**, we can use the same technique as in the proof of Lemma 4.2, yielding here that the problem is in **PPSPACE** (= probabilistic **PSPACE**). Ladner (1989) showed that #**PSPACE** equals the class of polynomial-space computable functions, from which it follows that **PPSPACE** = **PSPACE**.

Showing **PSPACE**-hardness is even easier than showing **PL**-hardness (as in the proof of Theorem 4.3), because here we deal with a deterministic class. We can consider the computation of a polynomial-space bounded machine M on input x as a directed graph with configurations as nodes and the transition relation between configurations as arcs. This graph is the skeleton of a compressed UMDP. Its nodes are the states, and each arc stands for transition probability 1 between the neighboured states. All other transition probabilities are 0. The only action means "perform a configuration transition." If an accepting configuration is reached, reward 1 is obtained; all other rewards are 0. Hence, the expected reward is greater than 0 if and only if an accepting configuration can be reached from the initial configuration. Because the transition probability function is essentially the polynomial-time predicate "given s, s', decide whether s' is a successor configuration of s", the circuit calculating it can be computed in polynomial time. Because the UMDP constructed is deterministic, each transition probability is either 0 or 1. Hence, the circuit constructed outputs all bits of the transition probability. Similar arguments apply for the circuit calculating the reward function. □

Note that we only used rewards 0 and 1 in the above hardness proof. Therefore, unlike the flat case, the same completeness result follows for POMDPs with nonnegative rewards.

Theorem 7.3 [3] *The long-term policy evaluation problem for compressed UMDPs with nonnegative rewards is* **PSPACE***-complete.*

7.2 Policy Existence

The results from Section 5.3 for short-term policy existence problems for flat POMDPs translate to compressed POMDPs. The complexity of the problems makes an exponential jump. The arguments used to prove the following results are direct translations of those used in the related proofs for flat POMDPs.

Theorem 7.4 [3] *The long-term time- or history-dependent policy existence problems for compressed MDPs are* **EXP***-complete.*

Proof. Given a compressed MDP $\mathcal{M}$, one can write down its flat representation $\mathcal{M}'$ in time $O(2^{|\mathcal{M}|})$. Now, the short-term time-dependent policy existence problem for $\mathcal{M}'$ is equivalent to the long-term time-dependent policy existence problem for $\mathcal{M}$, and takes time polynomial in $|\mathcal{M}'|$ by Theorem 5.7. Hence, the long-term time-dependent policy existence problem is in **EXP**.

To show **P**-hardness of the short-term time-dependent policy existence problem for MDPs, we used **P**-hardness of the circuit value problem. Since the *succinct circuit value problem* is hard for **EXP**, we obtain **EXP**-hardness of the long-term time-dependent policy existence problem for MDPs. □

Theorem 7.5 [3]

1. *The long-term stationary and time-dependent policy existence problems for compressed POMDPs are* **NEXP***-complete.*
2. *The long-term time-dependent policy existence problem for compressed UMDPs is* **NEXP***-complete.*
3. *The long-term history-dependent policy existence problem for compressed POMDPs is* **EXPSPACE***-complete.*

Note that **NEXP**-hardness also follows from Theorem 6.6.

Theorem 7.6 [3] *The long-term stationary policy existence problem for compressed UMDPs is* **PSPACE***-complete.*

Proof. A stationary policy for an UMDP consists of one action. Guessing an action and evaluating whether the UMDP's performance under this policy is positive can be performed in $\mathbf{NP}^{\mathbf{PSPACE}} = \mathbf{PSPACE}$.

Hardness for **PSPACE** follows as in the hardness part of the proof of Theorem 7.2. □

The situation is not different for POMDPs with nonnegative rewards.

Theorem 7.7 [3]

1. *The long-term stationary policy existence problem for compressed POMDPs with nonnegative rewards is* **NEXP***-complete.*

2. *The long-term time-dependent and history-dependent policy existence problems for compressed POMDPs with nonnegative rewards are* **PSPACE***-complete.*

3. *The long-term stationary, time-dependent, and history-dependent policy existence problems for compressed MDPs and for compressed UMDPs with nonnegative rewards are* **PSPACE***-complete.*

8 Partially Ordered Plans

The greatest challenge in using POMDPs as a basis for decision theoretic planning lies in discovering computationally tractable methods for the construction of optimal, approximate optimal or satisfactory policies. Of course, arbitrary decision problems are intractable – even producing approximately optimal policies is infeasible. A general technique for solving planning problems is *partial order planning*. In many cases, it can be determined that a particular action must be taken somewhere in the run of the process, but e.g. not necessarily in the last step. In this example, the search space for a policy is properly smaller than the set of all policies. Partial order planning algorithms proceed by stepwise making smaller the search space in which the optimal policy is searched. The search space is described in terms of a partial order, in which actions must be executed. We consider partial order planning with regard to short-term time-dependent policies. A time-dependent policy can be seen as a sequence $\pi_1, \pi_2, \ldots, \pi_n$ of stationary policies π_i, each called a *step*.

A *partially ordered plan* (sometimes called a "nonlinear" plan) is a set of steps and a partial order on this set. It expresses an entire family of time-dependent policies – all those that are consistent with the given partial order of steps and contain *all* steps. The partial order can be specified, for example, as a directed acyclic graph. The elements of the partial order are stationary policies, say $\pi_1, \pi_2, \ldots, \pi_n$ with some ordering $<$. A time-dependent policy consistent with the partial order is a sequence of *all* of these policies $\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_n}$ such that $\{i_1, i_2, \ldots, i_n\} = \{1, 2, \ldots, n\}$ and $\pi_{i_j} \not< \pi_{i_l}$ for every $j = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, j-1$. In Figure 20 a partially ordered plan is drawn that has 9 steps. Each step is a stationary policy for an unobservable POMDP mapping every observation either to 0 or to 1. There are 2^4 different time-dependent policies that are consistent with this plan.

Every consistent policy has a performance, and these need not all be the same. As such, we have a choice in defining the value for a partially ordered plan. We consider the optimistic, the average, and the median interpretations. Under the optimistic interpretation, a plan is accepted if it allows at least one consistent policy with positive performance.

Plan evaluation problem under the optimistic interpretation:

given a POMDP and a partially ordered plan,

decide whether the performance of a time-dependent policy consistent with the given plan is greater 0.

Notice that this problem has a flavour similar to the general policy existence problem. The difference is that the space in which a "good" policy is searched for, is described by the partially ordered plan. As it turns out, this restriction does not decrease the worst case complexity of the problem.

Under the average interpretation, a plan is accepted if the average of all performances of consistent policies is positive. In other words, if one randomly picks a policy consistent with an accepted

plan, then the expected performance is positive.

Plan evaluation problem under the average interpretation:

given a POMDP and a partially ordered plan,

decide whether the average performance of all time-dependent policies consistent with the given plan is greater 0.

Under the median interpretation, a plan is accepted if the median of all performances of consistent policies is positive. In other words, if one randomly picks a policy consistent with an accepted plan, then with probability $> \frac{1}{2}$ this policy has positive performance.

Plan evaluation problem under the median interpretation:

given a POMDP and a partially ordered plan,

decide whether the median performance of all time-dependent policies consistent with the given plan is greater 0.

8.1 Median and Average Performances

Before we consider partially ordered plans, we consider the average and median short-term performances of all stationary or time-dependent policies for a POMDP. This yields both the following decision problems.

Policy median problem:

given a POMDP and a policy type,

decide whether the median of the short-term performances of all policies of the specified type is greater 0.

This problem is intuitively harder than the policy existence problem, because we need to count the policies with positive performance. We show that it is complete for **PP** both for stationary and time-dependent policies. The second problem is called the *policy average problem.*

Policy average problem:

given a POMDP and a policy type,

decide whether the average short-term performance of all policies of the specified type is greater 0.

The latter turns out to be harder than the policy existence problem for stationary policies, but easier for time-dependent policies.

Theorem 8.1 *The stationary policy median problem for POMDPs is* **PP***-complete.*

Proof. We show that the problem is in **PP**. Given a POMDP $\mathcal{M}$, guess a stationary policy, calculate the performance of $\mathcal{M}$ under that policy, and accept if and only if the performance is greater than 0. This nondeterministic algorithm runs in polynomial time, because evaluating a given policy is in **PL** (Theorem 4.5), and hence in **P**. Because every policy is guessed with the same probability, this nondeterministic computation has more accepting than rejecting paths if and only if the median of all policy values of $\mathcal{M}$ is greater than 0.

To show **PP**-completeness, we give a reduction from the **PP**-complete set MAJSAT. A Boolean formula ϕ is in MAJSAT if and only if more than half of all assignments satisfy ϕ. Given ϕ, we construct a POMDP $\mathcal{M}(\phi)$ as in the proof of Theorem 5.5. Because of the bijection between stationary policies and assignments, we obtain that ϕ is in MAJSAT if and only if $\mathcal{M}(\phi)$ has performance 1 under more than half of all policies. Since $\mathcal{M}(\phi)$ has performance either 0 or 1 under any policy, we obtain that the median of $M(\phi)$ is greater than 0 if and only if ϕ is in MAJSAT. □

Notice that from the proof it follows that **PP**-completeness also holds for POMDPs with non-negative rewards. In case of the average problem, negative rewards seem to be necessary.

Theorem 8.2 *The stationary policy average problem for POMDPs is* **PP***-complete.*

Proof. In order to prove **PP**-hardness, we apply the same technique as in the proof of Theorem 8.1, but we change the reward function of the POMDP $\mathcal{M}(\phi)$ constructed from the formula ϕ in such a way that $\mathcal{M}$ has performance 1 under every stationary policy corresponding to a satisfying assignment, and performance -1 (instead of 0) under every other policy. Then the average of $\mathcal{M}(\phi)$ is greater than 0 if and only if ϕ is in MAJSAT.

Lemma 4.2 shows that for every POMDP $\mathcal{M}$ and policy π there exists a logspace polynomial-time nondeterministic computation such that the number of accepting paths minus the number of rejecting paths in this computation equals the performance of $\mathcal{M}$ under π multiplied by some constant dependent only on $\mathcal{M}$. To show that the average problem is in **PP**, instead of generating one computation path for each policy, we now generate such a computation tree for every policy in parallel. (Note that it is necessary to write down each policy, requiring polynomial rather than logarithmic space.) Then this whole computation has more accepting than rejecting paths if and only if the average performance of all policies is greater than 0. □

Theorem 8.3 *The time-dependent policy median problem for POMDPs is* **PP***-complete.*

Proof. Containment in **PP** can be shown using the same idea as in the stationary case (Theorem 8.1) – guess a policy and calculate its value. To show **PP**-hardness, we use a similar argument as for the stationary case. In the proof of Theorem 5.2, we showed how to reduce 3SAT to the existence problem for time-dependent policies for POMDPs by giving a transformation from formula ϕ to POMDP $\mathcal{M}(\phi)$ such that ϕ is satisfiable if and only if there exists a time-dependent policy under which $\mathcal{M}(\phi)$ has performance greater than 0. We have a bijective correspondence between time-dependent policies and assignments, such that $\mathcal{M}(\phi)$ has positive performance under a policy if and only if ϕ is satisfied by the correponding assignment. Therefore it follows that ϕ is in MAJSAT if and only if the median of $\mathcal{M}(\phi)$ under time-dependent policy is positive. □

The time-dependent policy average problem for POMDPs has complexity even lower than the existence problem – we locate it in **PL**. The difference between time-dependent and stationary policies is that it is not necessary in a trajectory for a time-dependent policy to remember previous choices. Thus, the space requirements drop dramatically.

Theorem 8.4 *The time-dependent policy average problem for POMDPs is* **PL***-complete.*

Proof. We start by showing that the problem is in **PL**. Let $\mathcal{M} = (S, s_0, A, O, t, o, r)$ be a POMDP. Imagine $\mathcal{M}$ as a graph with vertices labeled by states of $\mathcal{M}$ and edges labeled by actions between vertices corresponding to the state transition function of $\mathcal{M}$. Every path

$$\sigma_1 \xrightarrow{a_1} \sigma_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} \sigma_n$$

with $t(\sigma_i, a_i, \sigma_{i+1}) > 0$ $(i = 1, \ldots, n-1)$ is consistent with some time-dependent policy. Actually, every such path is consistent with exactly $|A|^{(|O|-1)\cdot(n-1)}$ time-dependent policies (since a time-dependent policy is simply a sequence of $n-1$ functions from O to A). Because every policy has equal probability, the average performance can be computed by computing the average performance of all paths as above. Let $av(i, m)$ denote the average performance of all paths through the graph for $\mathcal{M}$ that start in state i and make m steps. Clearly, $av(i, 0) = 0$. For $m > 0$, the average can inductively be be calculated as

$$av(i,m) = \frac{1}{|A|} \cdot \sum_{a \in A} \left(r(i,a) + \sum_{j \in S} t(i,a,j) \cdot av(j, m-1) \right) .$$

Because we are interested only in whether $av(i, m)$ is greater than 0, we can multiply both sides by $|A| \cdot 2^{|S| \cdot m}$ without changing the sign of the average. Then we obtain

$$\begin{aligned} & av(i,m) \cdot |A| \cdot 2^{|S| \cdot m} \; = \\ & \quad \sum_{a \in A} \left(2^{|S| \cdot m} \cdot r(i,a) + \sum_{j \in S} 2^{|S|} \cdot t(i,a,j) \cdot 2^{|S| \cdot (m-1)} \cdot av(j, m-1) \right) . \end{aligned}$$

Note that the right hand side of the equation now only deals with integer values. We argue that $av(i, m)$ is in **GapL**, i.e. the class of integer functions f for which exist logspace nondeterministic Turing machines N_f such that the number of accepting paths minus the number of rejecting paths of N_f on input x equals $f(x)$. The functions r and t are part of the input, and **GapL** is closed under all operations used on the right hand side of the equation (Allender and Ogihara 1996). Therefore it follows that $av(i, m) \cdot |A| \cdot 2^{|S| \cdot m}$ is computable in **GapL**. Thus the problem whether $av(i, m) \cdot |A| \cdot 2^{|S| \cdot m}$ is greater than 0 is decidable in **PL**. This follows using a characterization of **PL** by **GapL** from Allender and Ogihara (1996), also used in Section 4.1 (page 26).

Hardness for **PL** follows using the simulation technique as in the proof of Theorem 4.5, where rejecting computations have reward -1 and accepting computations have reward 1 (weighted depending on the length of the computation). □

8.2 Partially Ordered Plans

The plan evaluation problem for partially ordered plans is different from that of time-dependent policies. This is because a single partial not necessarily encodes *all* time-dependent policies. Evaluating a partially ordered plan involves figuring out the best (in case of optimistic interpretation) member, the median (for median interpretation) or the average (for average interpretation) of this combinatorial set.

Theorem 8.5 [1] *The plan evaluation problem for partially ordered plans under the optimistic interpretation is* **NP***-complete.*

Proof sketch. Membership in **NP** follows from the fact that we can guess any time-dependent policy consistent with the given partial order and accept if and only if the POMDP has positive performance under that policy. Remember that this evaluation can be performed in **PL** (Theorem 4.5), and therefore deterministically in polynomial time.

The **NP**-hardness proof is a variation of the construction used in Theorem 5.2. It uses the fact that for UMDPs, the set of all time-dependent policies can essentially be expressed as a partial order of size polynomial in the UMDP. The partially ordered plan to evaluate has the form given

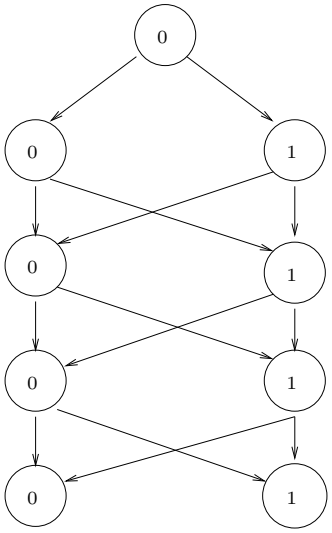


Figure 20: A partially ordered plan for a formula with 4 variables

in Figure 20. Every plan step is a policy that maps all observations either to action 0 or to action 1. The consistent total orders are of the form

$$0 \to b_1 \to \overline{b_1} \to b_2 \to \overline{b_2} \cdots \to b_n \to \overline{b_n}$$

where b_i is either 1 or 0, and $\overline{b_i} = 1 - b_i$. Each of the possible policies can be interpreted as an assignment to n Boolean variables by ignoring every second assignment action. The construction in the proof of Theorem 5.2 shows how to turn a CNF formula ϕ into a UMDP $\mathcal{M}(\phi)$, and it can easily be modified to ignore every second action. Thus, the best time-dependent policy consistent with the given partially ordered plan has performance > 0 if and only if it satisfies all clauses of ϕ if and only if ϕ is satisfiable. ■

Now we turn to the average interpretation. The time-dependent policy average problem had surprisingly low complexity (Theorem 8.4), because one did not need to guess a complete policy in order to calculate the average performance of all policies. This is not anymore the case if we consider time-dependent policies consistent to a partially ordered plan. As a consequence, the complexity of the plan evaluation problem is much higher.

Theorem 8.6 [1] *The plan evaluation problem for partially ordered plans under the average interpretation is* **PP***-complete.*

Proof. Under the average interpretation, we must decide whether the average value over all consistent time-dependent policies is greater than threshold 0. This can be decided in **PP** by guessing uniformly a time-dependent policy and checking its consistency with the given partially ordered plan in polynomial time. If the guessed policy is inconsistent, it must not contribute to the acceptance behaviour of the whole computation tree. Therefore, this computation path splits into one accepting and one rejecting path. Otherwise, the guessed policy is consistent. It can be evaluated in polynomial time (Theorem 4.5). Similar as in the proof of Theorem 8.2, every guessed consistent policy creates an according number of accepting paths, if its performance is positive, and an according number of rejecting paths, if its performance is negative. Hence, the whole

computation has more accepting than rejecting paths if and only if the average of all performances of all consistent policies is positive.

The **PP**-hardness of the problem is shown by a reduction from the **PP**-complete MAJSAT. Let ϕ be a formula in CNF. We show how to construct a UMDP $\mathcal{M}(\phi)$ and a partially ordered plan $P(\phi)$ such that $\phi \in$ MAJSAT if and only if the average performance of $\mathcal{M}(\phi)$ under a totally ordered plan consistent with $P(\phi)$ is greater than 0.

Let ϕ consist of the m clauses $C_1, \ldots, C_m$, which contain n variables $x_1, \ldots, x_n$. The UMDP $\mathcal{M}(\phi) = (S, s_0, A, O, t, o, r)$ has actions

$$A = \{\mathsf{assign}(i, b) \mid i \in \{1, \ldots, n\}, b \in \{0, 1\}\} \cup \{\mathsf{start}, \mathsf{check}, \mathsf{end}\}.$$

Action $\mathsf{assign}(i, b)$ will be interpreted as "assign b to x_i." The partially ordered plan $P(\phi)$ has plan steps

$$\begin{aligned} V &= \{\sigma(i, b, h) \mid i \in \{1, \ldots, n\}, b \in \{0, 1\}, h \in \{1, \ldots, m\}\} \\ &\cup \{\mathsf{start}, \mathsf{check}, \mathsf{end}\} \end{aligned}$$

of (constant) policies for the UMDP $\mathcal{M}(\phi)$ where

$$\sigma(i, b, h) = \mathsf{assign}(i, b).$$

The order E requires that a consistent time-dependent policy has start as the first and end as the last step. The steps in between are arbitrarily ordered. More formally,

$$E = \{(\mathsf{start}, q) \mid q \in V - \{\mathsf{start}, \mathsf{end}\}\} \cup \{(q, \mathsf{end}) \mid q \in V - \{\mathsf{start}, \mathsf{end}\}\}.$$

Now, we define how $\mathcal{M}(\phi)$ acts on a given time-dependent policy π consistent with $P(\phi)$. The UMDP $\mathcal{M}(\phi)$ consists of the cross product of the following polynomial-size deterministic UMDPs $\mathcal{M}_s$ and $\mathcal{M}_c$.

Before we describe $\mathcal{M}_s$ and $\mathcal{M}_c$ precisely, here are their intuitive definitions. $\mathcal{M}_s$ guarantees that $\mathcal{M}(\phi)$ has positive performance only under policies that have the form of an assignment to the n Boolean variables with the restriction that the assignment is repeated m times (for easy checking). $\mathcal{M}_c$ guarantees that $\mathcal{M}(\phi)$ has positive performance only under policies that correspond to satisfying assignments. The composite of these two processes is only satisfied by policies that correspond to satisfying assignments in the right form. We will now define these processes formally.

First, $\mathcal{M}_s$ checks whether the time-dependent policy matches the regular expression

$$\begin{aligned} &\mathsf{start}\ (\mathsf{assign}(1,0)^m | \mathsf{assign}(1,1)^m) \\ &\quad \cdots\ (\mathsf{assign}(n,0)^m | \mathsf{assign}(n,1)^m) \\ &\quad \mathsf{check}\ ((\mathsf{assign}(1,0) | \mathsf{assign}(1,1)) \cdots (\mathsf{assign}(n,0) | \mathsf{assign}(n,1)))^m \mathsf{end}. \end{aligned}$$

Note that the m here is a constant. Let "good" be the state reached by $\mathcal{M}_s$ if the plan matches that expression. Otherwise, the state reached is "bad". To clarify, the actions before check are there simply to "use up" the extra steps not used in specifying the assignment in the partially ordered plan.

Next, $\mathcal{M}_c$ checks whether the sequence of actions following the check action satisfies the clauses of ϕ in the following sense. Let $a_1 \cdots a_k$ be this sequence. $\mathcal{M}_c$ interprets each subsequence $a_{1+(j-1)\cdot n} \cdots a_{n+(j-1)\cdot n}$ with $a_{l+(j-1)\cdot m} = \mathsf{assign}(x, b_l)$ as assignment $b_1, \ldots, b_n$ to the variables $x_1, \ldots, x_n$, and checks whether this assignment satisfies clause C_j. If all single clauses are satisfied in this way, then $\mathcal{M}_c$ ends in a state "satisfied", and it ends in a state "unsatisfied" otherwise.

Note that $\mathcal{M}_s$ and $\mathcal{M}_c$ are defined so that they do not deal with the final end action. The rewards on all actions in $\mathcal{M}_s$ and $\mathcal{M}_c$ are 0. $\mathcal{M}(\phi)$ consists of the cross product of $\mathcal{M}_s$ and $\mathcal{M}_c$ with the transitions for action end as follows. If $\mathcal{M}(\phi)$ is in state (bad, q) for any state q of $\mathcal{M}_c$, then action end lets $\mathcal{M}(\phi)$ go to state "reject". This transition has reward 0. If $\mathcal{M}(\phi)$ is in state $(\text{good}, \text{satisfied})$, then $\mathcal{M}(\phi)$ under action end goes to state "accept". This transition has reward 1. Otherwise, $\mathcal{M}(\phi)$ is in state $(\text{good}, \text{unsatisfied})$ and goes under action end to state "reject". This transition has reward -1.

We analyze the behavior of $\mathcal{M}(\phi)$ under any policy π consistent with $P(\phi)$. If $\mathcal{M}_s$ under π reaches state "bad", then $\mathcal{M}(\phi)$ under π has performance 0. Now, consider a policy π under which $\mathcal{M}_s$ reaches the state "good" – called a *good* policy. Then π matches the above regular expression. Therefore, for every $i \in \{1, \ldots, m\}$ there exists $b_i \in \{0, 1\}$ such that all steps $\sigma(i, b_i, h)$ are between start and check. Thus, all steps between check and end are

$$\sigma(1, 1 - i_1, 1) \cdots \sigma(n, 1 - i_n, 1)\sigma(1, 1 - i_1, 2) \cdots \sigma(n, 1 - i_n, m)$$

Consequently, the sequence of actions defined by the labeling of these plan steps are

$$(\mathsf{assign}(1, i_1)\mathsf{assign}(2, i_2) \cdots \mathsf{assign}(n, i_n))^m.$$

This means, that $\mathcal{M}_c$ checks whether all clauses of ϕ are satisfied by the assignment $i_1 \cdots i_n$, i.e., $\mathcal{M}_c$ checks whether $i_1 \cdots i_n$ satisfies ϕ. Therefore, $\mathcal{M}(\phi)$ has reward 1 under a good policy π, if π represents a satisfying assignment, and reward -1 under a good policy otherwise.

Note that each assignment corresponds to exactly one good policy. Only good policies have performance other than 0. Therefore, the average performance over all good policies for $\mathcal{M}(\phi)$ is greater 0 if and only if there are more satisfying than unsatisfying assignments for ϕ, and the latter is equivalent to $\phi \in$ MAJSAT. □

Essentially the same technique is used to show the same completeness result for the median problem.

Theorem 8.7 *The plan evaluation problem for partially ordered plans under the median interpretation is* **PP***-complete.*

Proof sketch. The following algorithm shows that the problem is in **PP**. It guesses a policy. If it is not consistent, then the computation path splits into one accepting and one rejecting path. If the policy is consistent and has performance > 0, then this computation path accepts. If the policy is consistent and has performance ≤ 0, then this computation path rejects. Clearly, there are more accepting than rejecting computation paths if and only if the median of the performances of all consistent policies is greater than 0.

PP-hardness is shown by a reduction from the **PP**-complete MAJSAT similar to that in the proof of Theorem 8.6. The additional problem is that we cannot ignore the "bad" policies. Because there are much more bad than good policies, the median will be 0 independent on whether ϕ is in MAJSAT. Therefore, we add an extra action at the beginning of the process. This action is chosen from two possible choices. Dependent on this action, a bad policy either has performance -2 or it has performance 2. The performance of good policies is not influenced. Notice that there are as many policies with performance -2 as there are those with performance 2. Therefore, the median performance of all policies is greater 0 if and only if there are more good policies corresponding to satisfying assignments than there are good policies corresponding to unsatisfying assignments, i.e. $\phi \in$ MAJSAT. ■

9 Conclusion

Ja, mach nur einen Plan
Sei nur ein großes Licht!
Und mach dann noch 'nen zweiten Plan
Geh'n tun sie beide nicht.

Bertold Brecht: Dreigroschenoper

This work comprehensively explored the computational complexity of policy evaluation, policy existence, policy approximation and value approximation in partially-observable Markov decision processes. The results and the proof techniques are general enough to be applied also at other models of probabilistic processes (see Goldsmith, Littman, and Mundhenk (1997) and Littman, Goldsmith, and Mundhenk (1998)). One general lesson that can be drawn from the complexity results is that there is no simple relationship among the policy existence problems for stationary, time-dependent, and history-dependent policies. Although it is trivially true that if a good stationary policy exists, then good time- and history-dependent policies exist, it is not always the case that one of these problems is easier than the other.

The immediate contributions this work makes to the fields of control theory, economics, medicine, etc. are largely negative: it is unlikely that there are efficient algorithms for finding nearly optimal policies for the general POMDP problems.

However, there is good news in this bad news. Besides classifying the hardness of known hard problems, this work has useful corollaries. Once the corresponding decision or approximation problem is shown to be hard for a particular complexity class, the known approximation heuristics for equi-complex optimization problems can be applied. For instance, there are many **NP**-optimization heuristics (see the book of Hochbaum (1997) for a reasonable introduction) used in practice, and a growing body of heuristics for **PSPACE**-hard optimization problems.

Another major contribution of this work are natural problems that are complete for the class $\mathbf{NP}^{\mathbf{PP}}$. This class promises to be very useful to researchers in uncertainty in artificial intelligence because it captures the type of problems resulting from choosing ("guessing") a solution and then evaluating its probabilistic behavior. This is precisely the type of problem faced by planning algorithms in probabilistic domains, and captures important problems in other domains as well, such as constructing explanations in belief networks and designing robust communication networks. Before, those problems were simply classified as **NP**-hard and in **PSPACE**. With the advent of $\mathbf{NP}^{\mathbf{PP}}$-completeness proofs, algorithms have begun to be developed for such problems (Majercik and Littman 1998a; Majercik and Littman 1998b; Littman 1999b). The new and conceptually simple $\mathbf{NP}^{\mathbf{PP}}$-complete problem, E-MAJSAT, may be useful in further explorations in this direction. (See Littman (1999a) for experimental data on nondeterministically and probabilistically constrained formulas.) Now that we have explored $\mathbf{NP}^{\mathbf{PP}}$ as an important complexity class, practitioners are asked to find heuristics in order to manage its problems, and theorists are asked to investigate approximability of problems in $\mathbf{NP}^{\mathbf{PP}}$, as it was already done for **NP** and **PSPACE**.

There are a few problems that have not yet been categorized by completeness. However, the major open questions are of the form: What now?

There is a growing literature on heuristic solutions for POMDP (see Smallwood and Sondik (1973), Platzman (1977), Lovejoy (1991), Hauskrecht (1997), Cassandra (1998), Hansen (1998b), Lusena, Li, Sittinger, Wells, and Goldsmith (1999), Meuleau, Kim, Kaelbling, and Cassandra (1999), Peshkin, Meuleau, and Kaelbling (1999), for instance, besides those mentioned above).

Since these algorithms do not yield guaranteed optimal or near-optimal solutions, we leave a discussion of them to other sources. Recently, a variant of 2TBNs has been introduced using ADDs (arithmetic decision diagrams) instead of tables or trees (Hoey, St-Aubin, Hu, and Boutilier 1999). Actually, ADDs improve the performance of the standard algorithms. It is currently investigated how "natural" systems modeled as POMDPs can be decomposed in relatively small and independent subsystems whose problems can be solved using ADDs. Future will hopefully show that this approach is better than what Brecht calls "*zweiter Plan*".

References

Allender, E. and M. Ogihara (1996). Relationships among PL, #L, and the determinant. *RAIRO Theoretical Informatics and Applications 30*(1), 1–21.

Àlvarez, C. and B. Jenner (1993). A very hard log-space counting class. *Theoretical Computer Science 107*, 3–30.

Aoki, M. (1965). Optimal control of partially observed Markovian systems. *Journal of the Franklin Institute 280*, 367–386.

Astrom, K. (1965). Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications 10*, 174–205.

Balcázar, J., A. Lozano, and J. Torán (1992). The complexity of algorithmic problems on succinct instances. In R. Baeza-Yates and U. Manber (Eds.), *Computer Science*, pp. 351–377. Plenum Press.

Beauquier, D., D. Burago, and A. Slissenko (1995). On the complexity of finite memory policies for Markov decision processes. In *Math. Foundations of Computer Science*, Lecture Notes in Computer Science #969, pp. 191–200. Springer-Verlag.

Beigel, R., N. Reingold, and D. Spielman (1995). PP is closed under intersection. *Journal of Computer and System Sciences 50*, 191–202.

Bellman, R. E. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society 60*, 503–516.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Blondel, V. and J. Tsitsiklis (2000). A survey of computational complexity results in systems and control. *Automatica 36*(9), 1249–1274.

Blythe, J. (1999). Decision-theoretic planning. *AI Magazine 20*(2), 37–54.

Boutilier, C., T. Dean, and S. Hanks (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of AI Research 11*, 1–94.

Boutilier, C., R. Dearden, and M. Goldszmidt (1995). Exploiting structure in policy construction. In *14th International Conference on AI*.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence 69*, 165–204.

Cassandra, A. (1998). *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. Ph. D. thesis, Brown University.

Cassandra, A., L. Kaelbling, and M. L. Littman (1994). Acting optimally in partially observable stochastic domains. In *Proceedings of AAAI-94*.

Cassandra, A., L. Kaelbling, and M. L. Littman (1995). Efficient dynamic-programming updates in partially observable Markov decision processes. Technical Report CS-95-19, Brown University, Providence, Rhode Island.

Cassandra, A., M. L. Littman, and N. Zhang (1997). Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In D. Geiger and P. P. Shenoy (Eds.), *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 54–61. Morgan Kaufmann Publishers.

Dynkin, E. (1965). Controlled random sequences. *Theory and Probability Appl. 10*, 1–14.

Fenner, S., L. Fortnow, and S. Kurtz (1994). Gap-definable counting classes. *Journal of Computer and System Sciences 48*(1), 116–148.

Galperin, H. and A. Wigderson (1983). Succinct representation of graphs. *Information and Control 56*, 183–198.

Golabi, K., R. Kulkarni, and G. Way (1982). A statewide pavement management system. *Interfaces 12*, 5–21.

Goldsmith, J., M. L. Littman, and M. Mundhenk (1997). The complexity of plan existence and evaluation in probabilistic domains. In *Proc. 13th Conf. on Uncertainty in AI*. Morgan Kaufmann Publishers.

Hansen, E. (1998a). *Finite-Memory Control of Partially Observable Systems*. Ph. D. thesis, Dept. of Computer Science, University of Massachussets at Amherst.

Hansen, E. (1998b). Solving POMDPs by searching in policy space. In *Proceedings of AAAI Conference Uncertainty in AI*.

Hauskrecht, M. (1997). *Planning and Control in Stochastic Domains with Imperfect Information*. Ph. D. thesis, Massachusetts Institute of Technology.

Hochbaum, D. (1997). *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company.

Hoey, J., R. St-Aubin, A. Hu, and C. Boutilier (1999). SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 279–288.

Howard, R. (1960). *Dynamic Programming and Markov Processes*. MIT Press.

Jung, H. (1985). On probabilistic time and space. In *Proceedings 12th ICALP*, Lecture Notes in Computer Science #194, pp. 281–291. Springer-Verlag.

Ladner, R. (1989). Polynomial space counting problems. *SIAM Journal on Computing 18*, 1087–1097.

Laskey, K. B. and H. Prade (Eds.) (1999). *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, San Francisco.

Littman, M. L. (1994). Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, Lecture Notes in Computer Science. MIT Press.

Littman, M. L. (1996). *Algorithms for Sequential Decision Making*. Ph. D. thesis, Brown University.

Littman, M. L. (1997a). Probabilistic propositional planning: Representations and complexity. In *Proc. 14th National Conference on Artificial Intelligence*. AAAI Press/The MIT Press.

Littman, M. L. (1997b). Probabilistic propositional planning: Representations and complexity. In *Proc. 14th National Conference on AI*. AAAI Press / MIT Press.

Littman, M. L. (1999a). Initial experiments in probabilistic satisfiability. In *Proceedings of AAAI-99*.

Littman, M. L. (1999b). Initial experiments in stochastic satisfiability. In *Proceedings of Sixteenth National Conference on Artificial Intelligence*.

Littman, M. L., T. Dean, and L. Kaelbling (1995). On the complexity of solving Markov decision problems. In *Proceedings of the 11th Annual Conference on Uncertainty in AI*, pp. 394–402.

Littman, M. L., J. Goldsmith, and M. Mundhenk (1998). The computational complexity of probabilistic plan existence and evaluation. *The Journal of AI Research 9*, 1–36.

Lovejoy, W. (1991). A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research 28*, 47–66.

Lusena, C., J. Goldsmith, and M. Mundhenk (1998). Nonapproximability results for Markov decision processes. Technical Report 274-98, University of Kentucky Department of Computer Science.

Lusena, C., T. Li, S. Sittinger, C. Wells, and J. Goldsmith (1999). My brain is full: When more memory helps. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 374–381.

Madani, O., S. Hanks, and A. Condon (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

Majercik, M. and M. L. Littman (1998a). Maxplan: A new approach to probabilistic planning. In *Artificial Intelligence and Planning Systems*, pp. 86–93.

Majercik, S. M. and M. L. Littman (1998b). Using caching to solve larger probabilistic planning problems. In *Proceedings of Fifteenth National Conference on Artificial Intelligence*, pp. 954–959.

Meuleau, N., K.-E. Kim, L. P. Kaelbling, and A. R. Cassandra (1999). Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 417–426.

Monahan, G. (1982). A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Manag. Sci. 28*, 1–16.

Mundhenk, M. (2000). The complexity of optimal small policies. *Mathematics of Operations Research 25*(1), 118–129.

Mundhenk, M., J. Goldsmith, and E. Allender (1997). The complexity of the policy existence problem for partially-observable finite-horizon Markov decision processes. In *Proc. 25th Mathematical Foundations of Computer Sciences*, Lecture Notes in Computer Science #1295, pp. 129–138. Springer-Verlag.

Mundhenk, M., J. Goldsmith, C. Lusena, and E. Allender (2000). Complexity results for finite-horizon Markov decision process. *Journal of the ACM*. To Appear.

Papadimitriou, C. (1994). *Computational Complexity*. Addison-Wesley.

Papadimitriou, C. and J. Tsitsiklis (1987). The complexity of Markov decision processes. *Mathematics of Operations Research 12*(3), 441–450.

Papadimitriou, C. and M. Yannakakis (1986). A note on succinct representations of graphs. *Information and Control 71*, 181–185.

Parr, R. and S. Russell (1995). Approximating optimal policies for partially observable stochastic domains. In *Proceedings of IJCAI-95*.

Peshkin, L., N. Meuleau, and L. P. Kaelbling (1999). Learning policies with external memory. In *Proceedings of the Sixteenth International Conference on Machine Learning*.

Platzman, L. (1977). *Finite-memory Estimation and Control of Finite Probabilistic Systems*. Ph. D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Puterman, M. L. (1994). *Markov Decision Processes*. John Wiley & Sons, New York.

Pyeatt, L. (1999, July). *Integration of Partially Observable Markov Decision Processes and Reinforcement Learning for Simulated Robot Navigation*. Ph. D. thesis, Colorado State University.

Shamir, A. (1992, October). IP = PSPACE. *Journal of the ACM 39*(4), 869–877.

Simmons, R. and S. Koenig (1995). Probabilistic robot navigation in partially observable environments. In *Proceedings of IJCAI-95*.

Smallwood, R. and E. Sondik (1973). The optimal control of partially observed Markov processes over the finite horizon. *Operations Research 21*, 1071–1088.

Sondik, E. (1971). *The Optimal Control of Partially Observable Markov Processes*. Ph. D. thesis, Stanford University.

Streibel, C. (1965). Sufficient statistics in the optimal control of stochastic systems. *J. Math. Anal. Appl. 12*, 576–592.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. The MIT Press.

Toda, S. (1991). PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing 20*, 865–877.

Torán, J. (1991). Complexity classes defined by counting quantifiers. *Journal of the ACM 38*(3), 753–774.

Tseng, P. (1990, September). Solving h-horizon, stationary Markov decision problems in time proportional to $\log h$. *Operations Research Letters 9*(5), 287–297.

van der Gaag, L., S. Renooij, C. Witteman, B. Aleman, and B. Tall (1999). How to elicit many probabilities. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 647–654.

Vinay, V. (1991). Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proc. 6th Structure in Complexity Theory Conference*, pp. 270–284. IEEE.

Wagner, K. (1986). The complexity of combinatorial problems with succinct input representation. *Acta Informatica 23*, 325–356.

White, III, C. (1991). Partially observed Markov decision processes: A survey. *Annals of Operations Research 32*, 215–230.

Zhang, N. and W. Liu (1997). A model approximation scheme for planning in partially observable stochastic domains. *Journal of Artificial Intelligence Research 7*, 199–230.